IBM

Application System/400™

Binder SX21-9926-0

Digitized by the Internet Archive
in 2026

https://archive.org/details/manualsplus_12444

SC21-8078-0

AS/400™

Programming: Work Management Guide

AS/400™

SC21-8078-0

Programming: Work Management Guide

First Edition (June 1988)

This edition applies to Release 1 Modification Level 0 of the IBM Operating System/400 Licensed Program (Program 5728-SS1), and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or technical newsletters.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

The numbers at the bottom right of illustrations are publishing control numbers and are not part of the technical content of this manual.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to your IBM-approved remarketer.

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department 245, Rochester, Minnesota, U.S.A. 55901. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

AS/400 is a trademark of the International Business Machines Corporation.

© Copyright International Business Machines Corporation 1988

About This Manual

This manual describes work management – what it is and how to use it and the work management terms. This manual will help you set up your initial work management environment and change work management objects to meet your needs. Other topics that are covered are collecting performance data, tuning your performance, and gathering data on who is using the system and what resources they are using.

This manual may refer to products that are announced, but are not yet available.

Who Should Use This Manual

This manual is intended for the system or application programmer.

What You Should Know

Before using this manual, you should be familiar with general programming concepts and terminology, and have a general understanding of the overview of the AS/400 system and OS/400. You should also be familiar with the display stations and printers you are using.

How This Manual Is Organized

This manual is organized as follows:

Chapters 1 and 2 describe work management terms and how to set up and change your management environment to meet your needs.

Chapter 3 describes group jobs (which allow you to start many interactive jobs at one work station) and Attention key handling programs.

Chapter 4 gives examples of changing work management objects.

Chapter 5 describes some performance guidelines to help you tune your system.

Chapter 6 describes how to gather performance data and write an application to reduce the data into meaningful reports.

Chapter 7 describes the system values and how to use them to control or change the overall operation of the system.

Chapter 8 describes how to gather data to determine who is using the system and what resources they are using.

The last part of this manual contains a glossary and an index. Use the glossary to find the meaning of an unfamiliar term. Use the index to look up a topic and to see on which pages the topic is covered.

Related Online Information

The following online information is available on the AS/400 system. You can press the Help key a second time to see an explanation of how the online information works, including the index search function.

Help for Displays

You can press the Help key on any display to see information about the display. There are two types of help available:

General
Specific

General help explains the purpose of the display. General help appears if you press the Help key when the cursor is outside the areas for which specific help is available.

Specific help explains the field on which the cursor is positioned when you press the Help key. For example, it describes the choices available for a prompt. If a system message appears at the bottom of the display, position the cursor on the message and press the Help key to see information about the cause of the message and the appropriate action to take.

To exit the online information, press F3 (Exit). You return to the display on which you pressed the Help key.

Index Search

Index search allows you to specify the words or phrases you want to see information about. To use index search, press the Help key, then press F11 (Search index).

Help for Control Language Commands

To see prompts for parameters for a control language command, type the command, then press F4 (Prompt) instead of the Enter key.

Online Education

AS/400 online education provides tutorials on a wide variety of topics. To use the online education, press F13 (User support) on any system menu to show the User Support menu. Then select the option to use online education.

Question-and-Answer Function

The question-and-answer (Q & A) function provides answers to questions you may have about using the AS/400 system. To use the Q & A function, press F13 (User support) on any system menu to show the User Support menu. Then select the option to use the question-and-answer function.

Related Printed Information

You may want to refer to other AS/400 manuals for more specific information about a particular topic. The following is a list of the additional manuals and the information you may want to find in them:

- *Programming: Backup and Recovery Guide*, SC21-8079

  This manual provides the system programmer with information about the different media available to save and protect system data as well as a description of how to record changes made to data base files and how that information can be used for system recovery and activity report information.
- *Programming: Command Reference Summary*, SC21-8076

  This manual provides the system operator, system programmer, or system administrator with quick reference information about the structure of the AS/400 commands. This manual contains an alphabetic list of all AS/400 commands and a list, by command, of error messages the programmer can monitor for when writing programs.
- *Programming: Control Language Programmer's Guide*, SC21-8077

  This manual provides the application programmer or system programmer with a wide-ranging discussion of the AS/400 programming topics, including:
  - A general discussion of objects and libraries
  - CL programming, controlling flow and communicating between programs, working with objects in CL programs, and creating CL programs
  - Predefined and impromptu messages and message handling
  - How to define and create user-defined commands and menus
- *Programming: Control Language Reference*, SBOF-0481

  This manual provides the application programmer or system programmer with a description of the AS/400 control language (CL) commands. Each command is defined, including its syntax diagram, parameters, default values, and keywords.
- *Programming: Data Management Guide*, SC21-9658

  This manual provides the application programmer or system programmer with information about managing key aspects of the system, including:
  - Override and copy files.
  - Describe display, printer, tape, and diskette device files to the system.
  - Spooling, job queues, spooled output files, and output queues. This includes information on how to create jobs and output queues.
- *Programming: Performance Tools Guide*, SC21-8084

  This manual provides the system programmer with information about what performance management is, gives an overview of the tools, and tells how the tools can be used to help manage system performance. The manual gives instructions on how to approach the analysis of system performance and how to do system performance measurement, reporting, capacity planning, and application analysis.
- *Programming: Security Concepts and Planning*, SC21-8083

  This manual provides the system programmer (or someone who is assigned the responsibilities of a security officer) with information about general security concepts and planning for security on the system. It also includes information for all users about resource security.

- *Programming: System Reference Summary*, SC21-8104

  This manual provides the system operator or system programmer with quick reference information when working with the AS/400 system. This manual contains summaries of information (summary charts, system values, and DDS keywords) from the AS/400 group.
- *System Operations: Operator's Guide*, SC21-8082

  This manual provides the system operator or system administrator with information about how to use the system unit operator panel; send and receive messages; respond to error messages; start and stop the system; use the display station function keys; control devices; and also process and manage jobs on the system.
- *Communications: Advanced Program-to-Program Communications and Advanced Peer-to-Peer Networking User's Guide*, SC21-9598

  This manual is intended for the system programmer responsible for defining or using advanced peer-to-peer networking (APPN), and for writing application programs that use advanced program-to-program communications (APPC).
- *Communications: Asynchronous Communications Programmer's Guide*, SC21-9592

  This manual provides the application programmer or system programmer with the following information:
  - Description of asynchronous communications, including the types of communications lines and remote systems that are supported
  - Remote system generation of configuration requirements and the startup requirements needed for remote programs to communicate with the AS/400 system
  - Commands used by the AS/400 program to start a communications session
  - Programming considerations for the AS/400 system and for the remote system
- *Communications: Communications and Systems Management User's Guide* (to be available at a later date)

  This manual provides the system operator, system programmer, or system administrator with information for configuring the AS/400 system to use the remote management support (distributed host command facility), the change management support (distributed systems node executive), and the problem management support (alerts).
- *Communications: Distributed Data Management User's Guide*, SC21-9600

  This manual provides the application programmer or system programmer with information about remote file processing. It describes how to create a distributed data management (DDM) file, how to define a remote file to AS/400 DDM, what file utilities are supported through DDM, and the requirements of the AS/400 DDM as related to other systems.

- *Communications: Distribution Services Network Administrator's Guide*, SC21-9588

  This manual provides the system operator or system administrator with information about administering data communications applications on the AS/400 system. This guide may also be useful to a programmer who works with data communications functions on the AS/400 system. The reader is expected to be familiar with data communications concepts. The reader is also expected to use applicable AS/400 menus and displays or control language (CL) commands.
- *Communications: Programmer's Guide*, SC21-9590

  This manual provides the application programmer with the information needed to write application programs that use the AS/400 communications and the OS/400-ICF file. It also contains examples of communications programs and describes return codes.
- *Communications: User's Guide*, SC21-9601

  This manual provides the system operator, application programmer, system programmer, or system administrator with the following information:
  - Communications information that is common among the AS/400 communications support, such as:
    - Setting and changing communications values
    - Starting and stopping communications
  - Communications configuration information, such as defining lines, controllers, and devices
  - Information about defining and using display station pass-through
  - Information about the 3270 remote attachment
- *Communications: 3270 Device Emulation User's Guide*, SC21-9602

  This manual is intended for the display station operator and system programmer who use the binary synchronous communications (BSC) and system network architecture (SNA) 3270 device emulation. Device emulation is available for both displays and printers.

Contents

Chapter 1. Work Management Introduction 1-1
Overview of Work Management 1-1
Summary of Work Management Concepts and Terms 1-2
  Sources from Which Jobs Can Be Started 1-4
  Job Description Used for a Job 1-4
  Overview of Starting a Job 1-5
  Storage Pools in Which Routing Steps Run 1-6
  Starting and Ending a Subsystem 1-6
Characteristics of the Shipped System 1-7
  The Controlling Subsystem as Shipped by IBM 1-7
  Important System Values Shipped by IBM 1-8
  Subsystem Configurations Shipped by IBM 1-9
  Use of Shipped Objects for Interactive Jobs 1-10
  Use of Shipped Objects for Batch Jobs 1-18
  Use of Shipped Objects for Spooling Jobs 1-20
  Use of Shipped Objects for Communications Device Allocation 1-21
  Subsystem Summary 1-22

Chapter 2. Using Work Management Functions 2-1
Work Management Objects 2-3
  Subsystem Descriptions 2-3
  Changing the Sign-On Display File 2-11
  Job Descriptions 2-14
  Classes 2-15
  Job and Output Queues for Spooling 2-17
  Summary of Work Management Objects 2-20
Job Starting and Routing 2-23
  Interactive Job Illustrations 2-24
  End of Job Display 2-30
  Interactive Job Considerations 2-30
  Batch Job Illustrations 2-31
  Batch Job Considerations 2-35
  Submitting a Batch Job from Another Job 2-37
  Submitting a Job Using Parameters from a Display File 2-39
  Rerouting and Transferring Jobs 2-43
  Summary of Controls on Levels of Activity 2-45
Creating Your Own Subsystems 2-47
  Creating Another Controlling Subsystem 2-47
  Interactive Subsystem Example 2-48
Work Management Displays 2-50
Security Considerations with Job Descriptions 2-51
  Interactive Jobs 2-51
  Batch Jobs 2-51
  Autostart Jobs 2-52
  Communications 2-52
  Job Description Authority 2-52

Chapter 3. Group Jobs and Attention Key Handling Programs 3-1
Concepts for Group Jobs 3-1
  Starting, Handling, and Ending Group Jobs 3-3
  Relationship of Group Jobs to a Secondary Interactive Job 3-4
  Sample Application for Group Jobs 3-5

Effect of Call Level on Attention Key Status 3-9
  Conditions for Using the Attention Key 3-11
  Guidelines for Coding Attention Handling Programs 3-12
Designing an Attention Key Handling Program 3-13
  Ending Group Jobs 3-23
Performance Considerations 3-24

Chapter 4. Examples of Changing Work Management Objects 4-1
Controlling User Sign-Ons 4-1
  Using an Initial Program in a User Profile 4-2
  Using an Initial Menu in a User Profile 4-2
  Using Routing Data 4-2
  Using Request Data 4-3
  Using Double-Byte Request Data 4-3
  Preventing Request End or Sign-Off 4-3
Changing the Start-Up (QSTRUP) Program 4-4
Creating a Programmer Output Queue 4-4
Adding a Second Job Queue 4-5
Allowing More Batch Jobs to Run in QBATCH at a Time 4-5
Sending a Completion Message for Each Batch Job 4-5
Setting Up an Unattended Environment 4-6
Setting Up an Unattended Nighttime Environment 4-6
Calling a Special Recovery Program 4-8
Preventing Sign On Display from QINTER 4-8
Performing a Long-Running Function from a Work Station 4-8
Controlling a Group of Work Stations 4-9
Changing Priority and Time Slice 4-9
Mixing Double-Byte and Alphameric Display Stations 4-9
Time-Dependent Scheduling 4-11
  SNDTIMMSG Sample Command 4-11
  SBMTIMJOB Sample Command 4-12
  Starting the Request-Processing Job 4-15
  Security Considerations 4-16
  Recovery Considerations 4-16
  Creating the Objects 4-17
  Additional Request Commands 4-32

Chapter 5. Performance Guidelines 5-1
Performance Overview 5-2
  Job States 5-2
  Wait States 5-3
  System Objects/Process Access Groups (PAG) 5-5
  PAG Transfer 5-6
  Activity Levels and Ineligible Queue 5-7
  Time Slice 5-9
Performance Commands 5-10
  Work with System Status (WRKSYSSTS) Command 5-10
  Work with Active Jobs (WRKACTJOB) Command 5-13
Basic Tuning 5-14
  Initial Program Load (IPL) Performance Adjustments 5-14
  Initial Machine Pool Size 5-15
  Choosing Your Pool Configuration 5-17
  Adjustments to Pool Sizes and Activity Levels 5-19
  Adjusting Activity Levels 5-19
  Adjusting Pool Sizes 5-20
  Reviewing Your Performance 5-20

Specialized Tuning 5-20
  Specifying PURGE(*NO) for Interactive Jobs 5-21
  Separate Batch Work from *BASE 5-21
  Multiple Pools for Interactive Jobs 5-21
  Multiple Pools for Batch Jobs 5-22
  Summary 5-22

Chapter 6. Collecting Performance Data 6-1
Why to Collect Performance Data 6-1
Preparing to Collect Performance Data 6-1
How to Collect Performance Data 6-2
  Starting Performance Data Collection 6-2
  Ending Performance Data Collection 6-3
How Performance Data Collection Occurs 6-4
  Internal Data Collection Intervals 6-4
  When the Data Is Collected 6-5
  Notification of Performance Monitor Status 6-6
  Data Base File Management 6-6
Content of Performance Data Files 6-8

Chapter 7. System Values and Network Attributes 7-1
System Value Summary 7-1
  Date and Time System Values 7-2
  Editing System Values 7-2
  System Control System Values 7-3
  Library List System Values 7-5
  Allocation System Values 7-5
  Message and Logging System Values 7-5
  Storage System Values 7-6
Displaying a System Value 7-6
Changing a System Value 7-7
Detailed Description of System Values 7-8
  Date and Time System Values 7-8
  Editing System Values 7-10
  System Control System Values 7-11
  Library List System Values 7-22
  Allocation System Values 7-22
  Message and Logging System Values 7-25
  Storage System Values 7-26
Network Attributes 7-28

Chapter 8. Using Job Accounting 8-1
Job Accounting 8-1
  Resource Accounting 8-2
  Printer File Accounting 8-3
  Overview of Job Accounting 8-4
Job Accounting Operating Characteristics 8-6
Deciding Whether to Use Job Accounting 8-6
Accounting Codes 8-7
Resource Accounting Data 8-8
General Accounting Journal Information 8-8
  JB Accounting Journal Information 8-9
DP and SP Printer File Accounting Data 8-11
Batch Processing 8-13
Interactive Processing 8-13
System Job Processing 8-13

Setting Up Job Accounting 8-14
Accounting Journal 8-15
Analyzing Job Accounting Data 8-16
Security Considerations 8-16
Recovery Considerations 8-17
Converting Job Accounting Journal Entries 8-19
Job Accounting Approaches 8-20
  Validity Checking of Accounting Codes 8-20
  Controlling the Assignment of Accounting Codes in User Profiles 8-21

Glossary G-1

Index X-1

Chapter 1. Work Management Introduction

Work management supports the commands and internal functions necessary to control system operation and the daily work load on the system. In addition, work management contains the functions you need to distribute resources for your applications so that your system can handle your applications.

This chapter is divided into three main sections:

1. "Overview of Work Management," which gives a high-level overview of work management concepts and of IBM-supplied work management objects. The terms used in this section and the characteristics of the IBM-supplied objects are described in more detail in the remainder of the chapter.
2. "Summary of Work Management Concepts and Terms" on page 1-2, which contains the main concepts and terms you must know to understand work management. The work management functions relate closely to the basic performance and operating characteristics of your system.
3. "Characteristics of the Shipped System" on page 1-7, which describes the work management characteristics of the system as IBM ships it to you. The shipped system contains objects that let you do basic system functions in a simple and straightforward way.

Overview of Work Management

All the work done on the system is submitted through the work management functions. When the OS/400 is installed, it includes a work management environment that supports interactive, batch, and communications work, and spool processing. Because the work management functions can be used as they are installed, a complete understanding of the work management functions is not needed to do typical operations.

The OS/400 allows you to tailor this support or to create your own work management environment. To do this, you need an understanding of the work management concepts.

The following paragraphs give a high-level overview of the work management concepts and the objects supplied by IBM. These concepts and objects are described in more detail later in this chapter.

The basic types of jobs done on the system are interactive jobs, batch jobs, spooling jobs, and autostart jobs:

- An interactive job starts when a work station user signs on at a work station and ends when the work station user signs off.
- A batch job is started when the job is selected from the job queue. The job could have been read by a spooling reader and placed on the job queue, or it could have been submitted to the job queue from another job.

  A job started by a communications program start request from another machine is a communications batch job. Job queues are not used when starting a communications batch job.

- Spooling functions are available for both input and output. For input spooling, a system program called a reader transfers jobs from an input device (diskette or data base file) to a job queue. For output spooling, the system places output records produced by a program in a spooled output file. These files are later written to external devices (printers or diskette) by system jobs called writers. Spooling jobs are started when a start reader or start writer command is specified. The *Data Management Guide* contains more information on spooling.
- Autostart jobs can be used to automatically start jobs that perform repetitive work or one-time initialization-type work. Autostart jobs are associated with a particular subsystem, and each time the subsystem is started the autostart jobs associated with it are started.

On the AS/400[1] system, all user jobs operate in an environment called a *subsystem*. A group of jobs with common characteristics can be controlled independently of other jobs, if they are placed in the same subsystem. Subsystems can be easily started and ended as needed to support the work being done and to maintain the performance characteristics you desire. One subsystem, called a *controlling subsystem*, is automatically started during the initial program load (IPL). (For information on how to do an IPL, see the *Operator's Guide*.)

To use the system, each user must sign on with their user profile name. A user profile is an object that represents a particular user or group of users to the OS/400 and identifies which objects and functions the user has authority to. (The *Security Concepts and Planning* contains more information about user profiles.)

The following are user profiles supplied by IBM that are most widely used:

- QSYSOPR, which is used by the system operator
- QPGMR, which is used by a programmer for online programming
- QUSER, which is used by a work station user
- QSECOFR, which is used by the security officer to create new user profiles for users and groups of users to give better control of the system

Summary of Work Management Concepts and Terms

Figure 1-1 on page 1-3 shows the structure provided by and managed through work management. The referenced numbers refer to descriptions following the figure.

---

[1] AS/400 is a trademark of the International Business Machines Corporation.

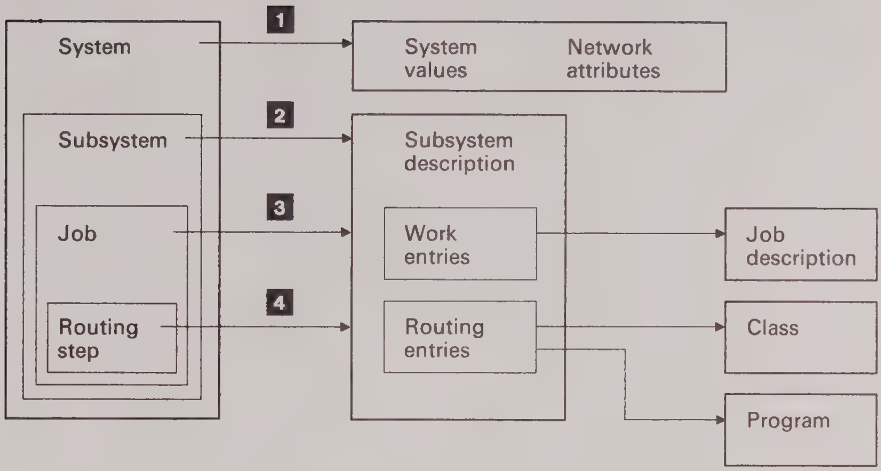


Figure 1-1. Work Management Structure

1 *System values and network attributes* are control values that affect how the system operates. Network attributes apply to a network of systems, not just a single system. Examples of system values are date, time, name of the controlling subsystem description, and maximum activity level of the system. Examples of network attributes are alert status, system name, and default message queue used in object distribution. You can change these values to change system operation. For a description of these values, including their initial values, see Chapter 7, "System Values and Network Attributes."

2 A subsystem is a single, predefined operating environment through which the system coordinates the work flow and resource use. The system can contain several subsystems, all operating independently of each other. Subsystems share resources. The run-time characteristics of a subsystem are defined in an object called a *subsystem description*. It defines what jobs can run within a subsystem and how they can be run. The subsystem takes on the simple name of the subsystem description. You can use a subsystem description supplied with your system (with or without making changes to it), or you can create your own subsystem descriptions. (See "Subsystem Descriptions" on page 2-3 for more information.)

3 Each piece of work run in a subsystem is called a *job*. Each job is a single, identifiable sequence of processing actions that represent a single use of the system. In a subsystem description, *work entries* are defined to identify the sources from which jobs can be started for running in that subsystem. Regardless of how a job is started, it must use a *job description*. A job description is an object in which most of the attributes of a job, such as default library list, message logging level, job queue, routing data, default printer, output scheduling priority, and user profile, are predefined and stored.

4 Jobs are processed in a subsystem as one or more consecutive *routing steps*. A routing step is the processing done as a result of calling the program specified in the subsystem description's *routing entry*. When a job is started, the correct routing entry is selected by means of *routing data*. The routing data can be extracted from the job description associated with the job, or specified when a batch job is placed on a job queue. In addition, a routing entry refers to another object called a *class* that contains parameters specifying the run-time environment for the routing step.

Sources from Which Jobs Can Be Started

In a subsystem description, *work entries* are defined to identify the sources from which jobs can be started for running in that subsystem. Any or all types of work entries can be specified in a single subsystem description. The types of work entries are:

- Autostart job entry. An autostart job entry specifies a job that is automatically started when the subsystem is started. An autostart job can be used to do initialization functions for an application or to give centralized service functions for other jobs in the subsystem. An autostart job in the controlling subsystem can be used to bring up other subsystems (as the IBM-supplied controlling subsystem does).
- Work station entry. A work station entry specifies one or a group of work stations from which interactive jobs can be started. An interactive job starts when a work station user signs on, and ends when the user signs off.
- Job queue entry. Each job queue entry specifies one of the job queues from which the subsystem can select batch jobs. A batch job is a command or group of commands that can run independently of a user at a work station. Jobs are placed on a job queue by a spooling reader, or by any of the following commands:
  - Submit Job (SBMJOB)
  - Submit Data Base Jobs (SBMDBJOB)
  - Submit Diskette Jobs (SBMDKTJOB)
  - Transfer Job (TFRJOB)
  - Transfer Batch Job (TFRBCHJOB)
- Communications entry. A communications entry specifies one or a group of communications device descriptions by name or type, or a remote location name from which communicating batch jobs may be started. For more information on communications functions, see the *Communications User's Guide*. The program on the system which is to be started from another system is specified in a routing entry or in the routing data for the job. Once the job is started, it is considered to be a batch job for accounting and identification purposes.

  For more information, see "Work Entries" on page 2-7.

Job Description Used for a Job

The job description for an interactive, autostart, or communications job is specified in the work entry from which the job is started. For a batch job, the job description is specified on the Batch Job (BCHJOB) or SBMJOB command used to submit the job to the system. The job attributes specified in the job description can be overridden when the job is placed on a job queue by the BCHJOB or SBMJOB command. Some job attributes can be changed while a job is running using the Change Job (CHGJOB) command. (For more information, see "Job Descriptions" on page 2-14.)

Overview of Starting a Job

Figure 1-2 shows the relationship of the parts of a subsystem description in starting a routing step.

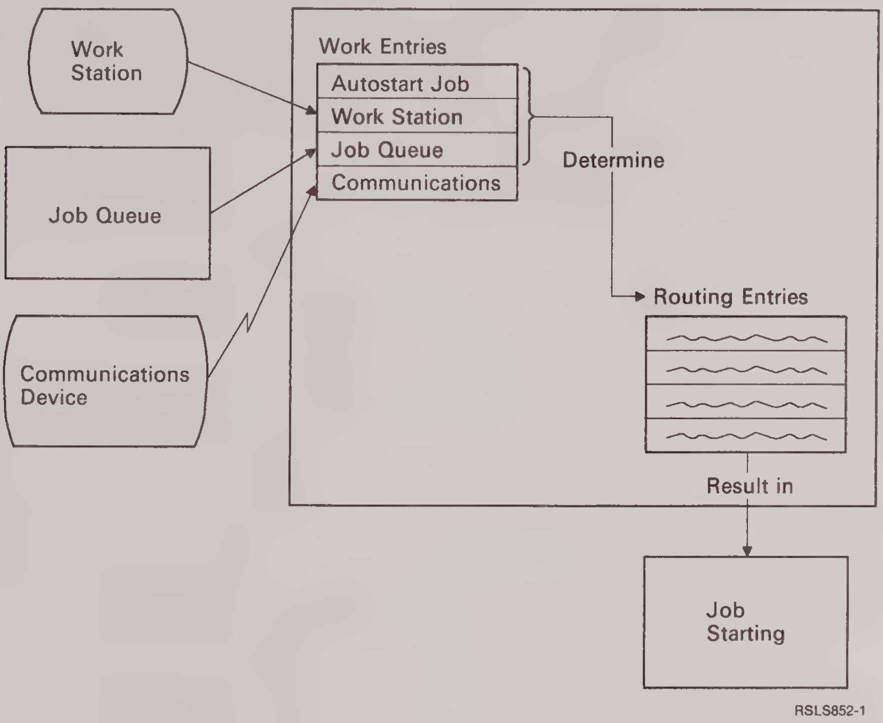


Figure 1-2. Starting a Job

Of primary importance in starting a job is routing data. *Routing data* gives the information needed to scan the routing entries in a subsystem description to determine how a routing step is to be started. For batch jobs, routing data comes from the job description or from the routing data (RTGDTA) parameter on the BCHJOB or SBMJOB command. For interactive jobs, routing data comes from the job description. For transferred or rerouted jobs (either batch or interactive), routing data comes from the TFRJOB, TFRBCHJOB, or Reroute Job (RRTJOB) command. For a communications job, the routing data is created by the subsystem based on information received in the program start request.

A routing entry contains values that control the selection of which routing entry is to start the routing step: sequence number, comparison value, and position at which the compare is to start. Three other values define the running environment for the routing step: program name, class name, and storage pool identifier.

The sequence number defines the order in which the routing entries are scanned and can be used as the identifier of the routing entry. The comparison value specifies data that is compared with routing data to determine which routing entry to use. (The routing entry also specifies the starting position for the comparison.) The routing data is compared with the comparison value of each routing entry in sequence number order until a match is found.

When a routing entry is found with a comparison value that matches the routing data, a routing step is started and the program specified in the routing entry is called. The running control values in the class associated with the routing entry are used for the routing step, and the routing step runs in the storage pool specified in the routing entry. (For more information, see "Routing Entries" on page 2-12.)

Storage Pools in Which Routing Steps Run

Storage pools are defined in subsystem descriptions. They give a means of reserving a certain amount of main storage for a particular use when a subsystem is started. More than one job in a subsystem can run in the same storage pool.

The main storage in your system is divided as follows:

- A machine storage pool used by the machine and specific highly shared OS/400 programs.
- A shared storage pool (the base storage pool) that can be shared across subsystems and in which many OS/400 functions run. (All the subsystems supplied by IBM share the base storage pool.)

  Note: QSPL is shipped using *BASE. QSPL and QBASE or QINTER will be changed to use a separate pool if performance adjustment at IPL (QPFRADJ) is used.
- As many as 14 additional storage pools that can be allocated for use in active subsystems.

A storage pool does not have to be large enough to contain your programs. The machine manages the transfer of data (and programs) into the storage pool as they are needed. If data is already in main storage, it can be referred to by a routing step independent of which storage pool it is in. However, if needed data does not exist in any storage pool, it is brought into the storage pool for the routing step that referred to it. As data is transferred into a storage pool, other data is displaced and, if changed, is automatically recorded in auxiliary storage. The storage pool size should be large enough to keep data transfers at a reasonable level as the rate affects performance.

The *activity level* of a storage pool is the number of jobs that can run at the same time in a storage pool. The machine manages the control of this level. Often during processing in a job, a program waits for a system resource or a response from a work station user. During such waits, a job gives up its use of the storage pool in order that another job that is ready to be processed can take its place.

More than one job can be active at the same time in a storage pool because the processing for a job can be briefly interrupted while needed data is gotten from auxiliary storage. During this delay, which is usually short, another job can run. Using the activity level, the machine can process a large number of jobs in a storage pool and, at the same time, hold the level of contention to the limit you specify. (For more information, see "Storage Pool Definitions" on page 2-4.)

Starting and Ending a Subsystem

A subsystem must be active before work can be done in it. During an IPL, a controlling subsystem is also started. This controlling subsystem controls the console. The IBM-supplied controlling subsystems (QBASE and QCTL) contain an autostart job that starts the other subsystems in their configuration. For any other subsystem to be active, it must be started using the Start Subsystem (STRSBS) command. The system operator usually has authority to this command and the End Subsystem (ENDSBS) command and, therefore, the operator can control what subsystems are active in the system.

Characteristics of the Shipped System

The system is shipped with a predefined set of work management objects that can be used immediately for processing. See the section on IBM-supplied objects in the *System Reference Summary* for a detailed description of these objects. You can continue to use these objects or you can create your own.

All subsystem descriptions are shipped with *USE authority given to all users of the system. This means that any user profile can run a job in any of the subsystems defined by the shipped subsystem descriptions. However, anyone changing a subsystem description must be running under a user profile with *CHANGE authority for the subsystem description.

The user profiles shipped with the system are designed for the various types of system users. For example, normal programmer functions are authorized to the QPGMR user profile; security officer functions are authorized to the QSECOFR user profile; and system operator functions are authorized to the QSYSOPR user profile. The user profile associated with a job determines what system functions can be done by that job.

These user profiles and subsystem descriptions are set up in order that any of several IBM-supplied programs such as QSYS/QCMD are called to support interactive jobs.

The program QSYS/QCMD processes CL commands specified interactively or in batch jobs. It is called during the running of most jobs that run on the system as it is shipped. An initial program can be specified in a user profile. If a job is routed to QSYS/QCMD and the user profile under which the job is running has an initial program, QSYS/QCMD calls the initial program. If no initial program is specified, QSYS/QCMD displays the initial menu.

Similarly, there is a program that displays a menu for the system operator (QSYSOPR). The System menu is displayed for the QSYSOPR user profile as a result of running the initial menu system.

The section on IBM-supplied objects in the *System Reference Summary* has complete descriptions of the shipped objects.

The Controlling Subsystem as Shipped by IBM

The controlling subsystem is the subsystem description name specified in the system value QCTLSBSD. When your system is shipped, QGPL/QBASE is specified in QCTLSBSD as the controlling subsystem. QGPL/QBASE specifies that all work stations are available for sign-on. The console is the only work station that can be used during an attended IPL. Therefore, if you assign another subsystem as the controlling subsystem, you must have a work station entry for the console in that subsystem description. This work station entry should specify that the subsystem display the Sign On display (specify WRKSTNTYPE(*CONS) AT(*SIGNON) on the Add Work Station Entry (ADDWSE) command).

The *Operator's Guide* contains a complete description of how to perform an IPL.

Important System Values Shipped by IBM

The system is shipped with many system values that control different aspects of system operation. The following are some of the most significant ones (see Chapter 7, "System Values and Network Attributes" for a full description of all system values):

QAUTOCFG: Controls whether the system should automatically create device descriptions for devices such as work stations that you attach to your system. The default is to perform automatic configuration. If you do not want the system to do this for you, enter the following command:

```
CHGSYSVAL SYSVAL(QAUTOCFG) VALUE('0')
```

QCTLSBSD: Specifies the name of the controlling subsystem description. This is used to automatically start the controlling subsystem during every IPL. The default is to use the QBASE shipped subsystem description. You can change this to specify a different subsystem description, and it will be used on the next IPL. For example, if you want to use the QCTL shipped subsystem description for the controlling subsystem, enter the following command (see the next section for more information about the shipped subsystem descriptions):

```
CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTL QGPL')
```

QDEVNAMING: Specifies what type of device names are to be assigned whenever the system automatically creates device descriptions. This is primarily for automatic configuration, but also applies to the device description created for the console. The default is to use names like DSP01 for work stations and PRT01 for printers. If you want to have names like W1 for work stations and P1 for printers, enter the following command:

```
CHGSYSVAL SYSVAL(QIPLTYPE) VALUE('1')
```

QIPLTYPE: Specifies how IPL is to occur. The default is that the IPL will be unattended, meaning that once you start the IPL and the key is not in the MANUAL position, all the system activity will be started without any operator intervention. If you want the operator to have a chance to select any special functions during the IPL or change certain system characteristics, set the Keylock switch to Manual or enter the following command:

```
CHGSYSVAL SYSVAL(QIPLTYPE) VALUE('1')
```

This causes the IPL to be attended, meaning that the normal system activity is not started until after the operator has responded to a series of displays (see the *Operator's Guide* for more information about the function of these displays). If the Keylock switch is set to Manual, the IPL will be attended regardless of this system value.

QSECURITY: Specifies the type of security enforced by the system. The default is to not require passwords or authority. See Chapter 7, "System Values and Network Attributes" or the *Security Concepts and Planning* for more information.:

QSPCENV: Specifies whether users by default are to run in a special environment. The default is to not run in a special environment. This can be overridden in each user profile. If you want all users by default to be running in the System/36 environment, enter the following command:

```
CHGSYSVAL SYSVAL(QSPCENV) VALUE('*S36')
```

Subsystem Configurations Shipped by IBM

Two complete subsystem configurations are supplied by IBM and can be used without changing. The one the system uses when you IPL is controlled by the controlling subsystem description system value (QCTLSBSD) which defaults to 'QBASE QGPL'. This default configuration consists of the following subsystem descriptions in library QGPL:

- QBASE: This is the controlling subsystem. It supports interactive, batch, and communications jobs. It has an autostart job which automatically starts the QSPL subsystem.
- QSPL: This is the spool subsystem. It supports reader and writer jobs.

The other more complex configuration, which is supplied by IBM, consists of the following subsystem descriptions in library QGPL:

- QCTL: This is the controlling subsystem. It only supports signing on at the console. It has an autostart job which automatically starts the QINTER, QBATCH, QCMN, and QSPL subsystem.
- QINTER: This supports all interactive jobs (except at the console).
- QBATCH: This supports all batch jobs.
- QCMN: This supports all communications jobs.
- QSPL: This is the spool subsystem. It supports reader and writer jobs.

The QBASE default configuration gives the ability to run all the same functions that you can run with the QCTL configuration and is easier to manage because it consists of fewer subsystems.

The QCTL configuration allows for more granular control over your system operations by dividing the system activity into different subsystems based on the type of activity. For example, if you want to run batch jobs over the weekend or overnight but do not want anyone to be able to sign on (except at the console), you can easily do that with the QCTL configuration by simply ending the QINTER subsystem. Another reason you might want to end one subsystem without affecting others would be to change some of the subsystem description shipped attributes without an IPL. Many of these cannot be changed while the subsystem is active. If you need to change one of these attributes and are using the QBASE configuration, you would need to make a copy of QBASE with your changes, change the QCTLSBSD system value, and then IPL the system.

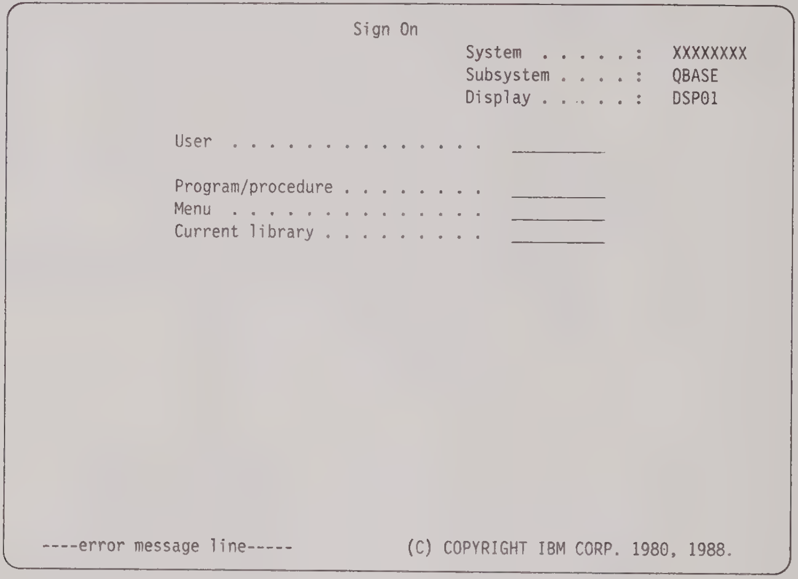
If you are considering creating your own subsystem configuration, you may also find that it is easier to use the QCTL configuration as a starting point than the QBASE configuration.

If you want to use the IBM-supplied QCTL subsystem configuration, enter the following command, which will take effect on the next IPL:

```
CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTL QGPL')
```

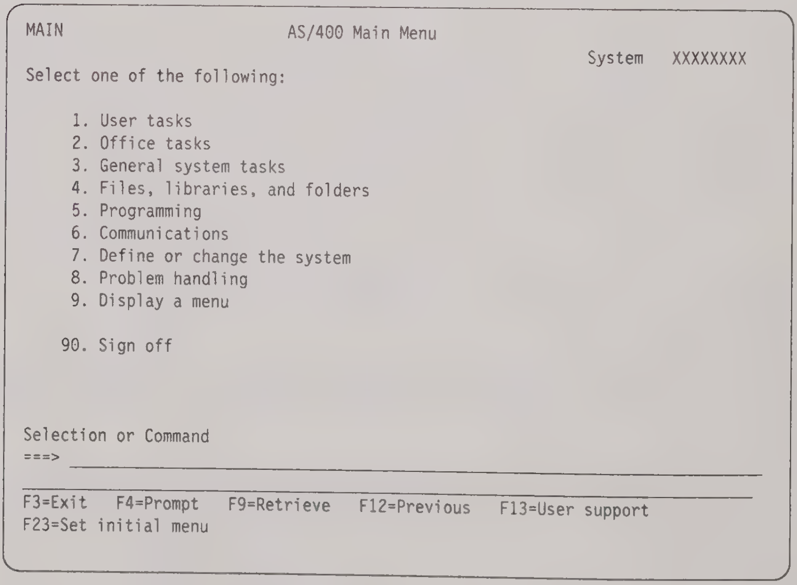
Use of Shipped Objects for Interactive Jobs

Figure 1-3 is an example of what the work station user sees at the work station. The use of the shipped objects by the OS/400 is not obvious to the work station user. For this example, the OS/400 and the subsystem QBASE have been started. It also assumes that the work station user has an initial menu of MAIN and no initial program is called that displays a menu.



```
                                Sign On
                                           System  . . . . . :   XXXXXXXX
                                           Subsystem . . . . :   QBASE
                                           Display . . . . . :   DSP01

                User  . . . . . . . . . . . . . .      __________

                Program/procedure . . . . . . . .      __________
                Menu  . . . . . . . . . . . . . .      __________
                Current library . . . . . . . . .      __________









  ----error message line-----             (C) COPYRIGHT IBM CORP. 1980, 1988.
```

↓ User enters user name and presses the Enter key.



```
 MAIN                          AS/400 Main Menu
                                                          System    XXXXXXXX
 Select one of the following:

      1. User tasks
      2. Office tasks
      3. General system tasks
      4. Files, libraries, and folders
      5. Programming
      6. Communications
      7. Define or change the system
      8. Problem handling
      9. Display a menu

     90. Sign off



 Selection or Command
 ===> ____________________________________________________________________

 F3=Exit   F4=Prompt   F9=Retrieve   F12=Previous   F13=User support
 F23=Set initial menu
```

Figure 1-3. Work Station User Signs on the AS/400 system

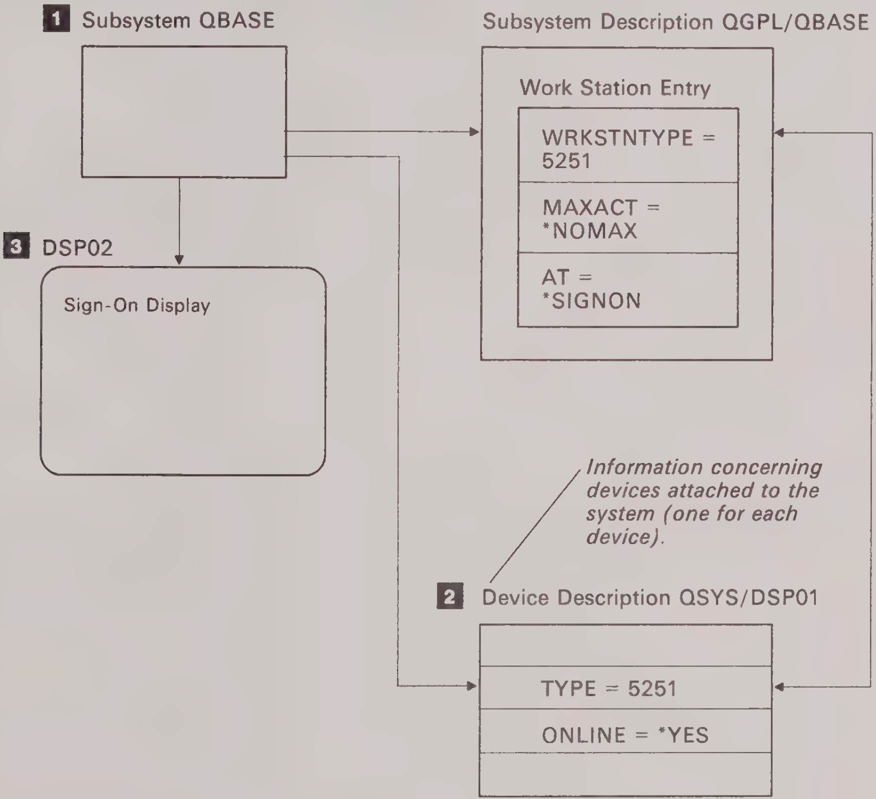
The first thing on the display is the Sign On display. If the work station user enters a user name and presses the Enter key, the AS/400 Main Menu is displayed. The system default does not require a password.

The work station user can then select functions from this menu. When the sign-off option is selected, the interactive job ends and the new Sign On display is displayed.

You can change the QSECURITY system value to require that users enter a valid password. See Chapter 7, "System Values and Network Attributes" or *Security Concepts and Planning* for more information on requiring passwords.

The next few pages describe how work management uses the objects previously discussed from the time a subsystem is started to when the Main Menu is shown. Figure 1-4 through Figure 1-7 on page 1-15 illustrate the steps needed.

Figure 1-4 summarizes what leads up to the display of the Sign On display.


1 Subsystem QBASE
Subsystem Description QGPL/QBASE
Work Station Entry
WRKSTNTYPE = 5251
MAXACT = *NOMAX
AT = *SIGNON
3 DSP02
Sign-On Display
Information concerning devices attached to the system (one for each device).
2 Device Description QSYS/DSP01
TYPE = 5251
ONLINE = *YES

1 Subsystem QBASE is started.

2 Work station devices are allocated.

3 Sign On display is shown.

RSLS853-2

Figure 1-4. Subsystem Activity Leading Up to the Sign On Display

The work station entries in the subsystem description QGPL/QBASE specify that jobs can be accepted from all work stations. When the QBASE subsystem is started, all online work stations are allocated to the subsystem (AT(*SIGNON) parameter) and the Sign On display is shown on each of the work stations.

Figure 1-5 shows what normally occurs when a programmer signs on to the system as shipped.

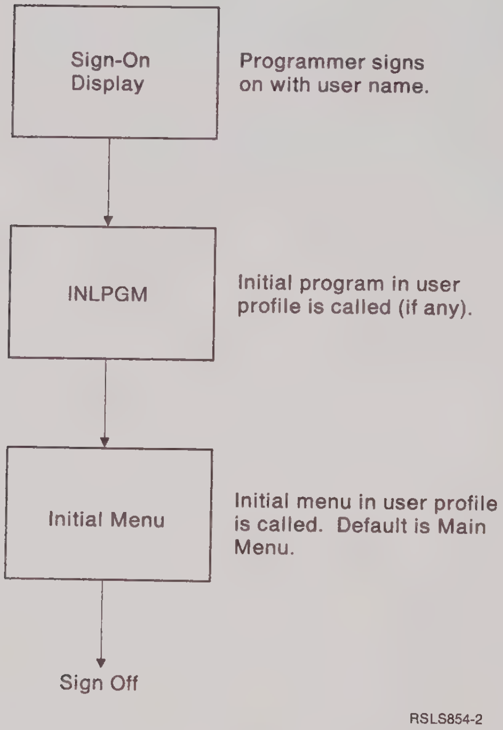
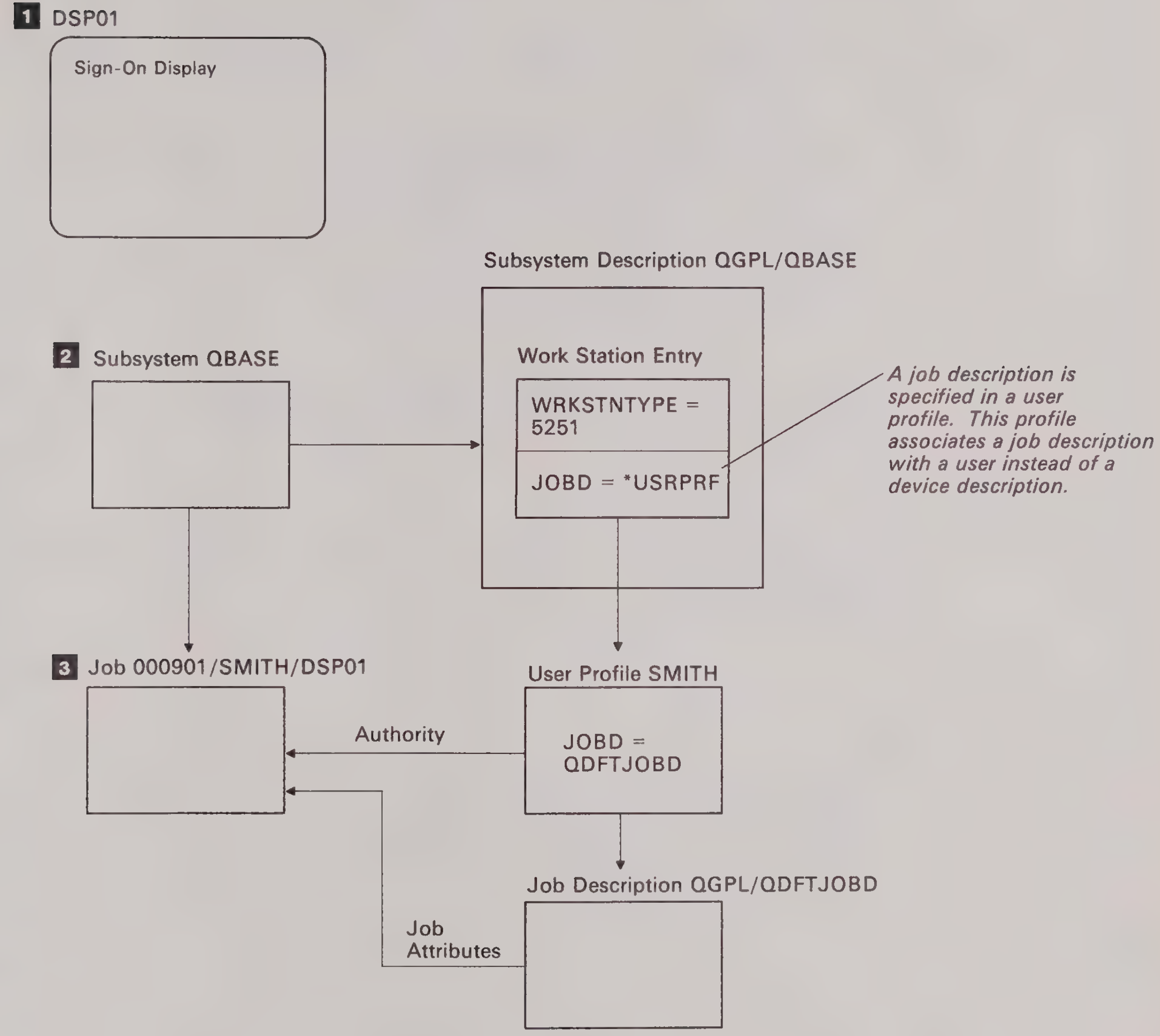

Sign-On Display
Programmer signs on with user name.
INLPGM
Initial program in user profile is called (if any).
Initial Menu
Initial menu in user profile is called. Default is Main Menu.
Sign Off
RSLS854-2

Figure 1-5. Programmer Signs on the AS/400 system

The programmer types in his user profile name and presses the Enter key. The Main Menu appears because the programmer does not have an initial program. The programmer can then specify commands or options from the menu to request functions of the system.

Figure 1-6 shows what happens when user SMITH types his user name and presses the Enter key.


1 DSP01
Sign-On Display
Subsystem Description QGPL/QBASE
2 Subsystem QBASE
Work Station Entry
WRKSTNTYPE = 5251
JOBD = *USRPRF
A job description is specified in a user profile. This profile associates a job description with a user instead of a device description.
3 Job 000901/SMITH/DSP01
User Profile SMITH
Authority
JOBD = QDFTJOBD
Job Description QGPL/QDFTJOBD
Job Attributes

1 User SMITH types his user name and presses the Enter key.

2 Subsystem detects a sign-on attempt.

3 Job is started.

RSLS880-1

Figure 1-6. Subsystem Activity Leading Up to the Creation of a Job

To make it easier to control and identify jobs on the system, each job has a unique qualified job name. The *qualified job name* consists of three parts: the job name, the user name, and the job number.

- For interactive jobs, the *job name* is the same as the name of the work station you signed on to. For batch jobs you can specify your own job name. The job name can be up to 10 characters long.
- The *user name* is the name of the user profile under which the job is running. For interactive jobs, the user name is the name you entered in the user field on the sign-on display. For batch jobs you can specify the user profile under which the batch job is to run. The user name can be up to 10 characters long.
- The *job number* is assigned by the system so that you can always uniquely identify any job, even if more than one has the same job name and user name. The job number is always 6 numeric digits.

The syntax for qualified job names is similar to qualified names for objects. For example, if the job name is DSP01, the user is QPGMR, and the job number is 000578, the qualified job name would be entered on the Work with Job (WRKJOB) command as follows:

```
WRKJOB JOB(000578/QPGMR/DSP01)
```

Another similarity to object names is that you do not need to specify all of the qualifiers. For example you could specify the following:

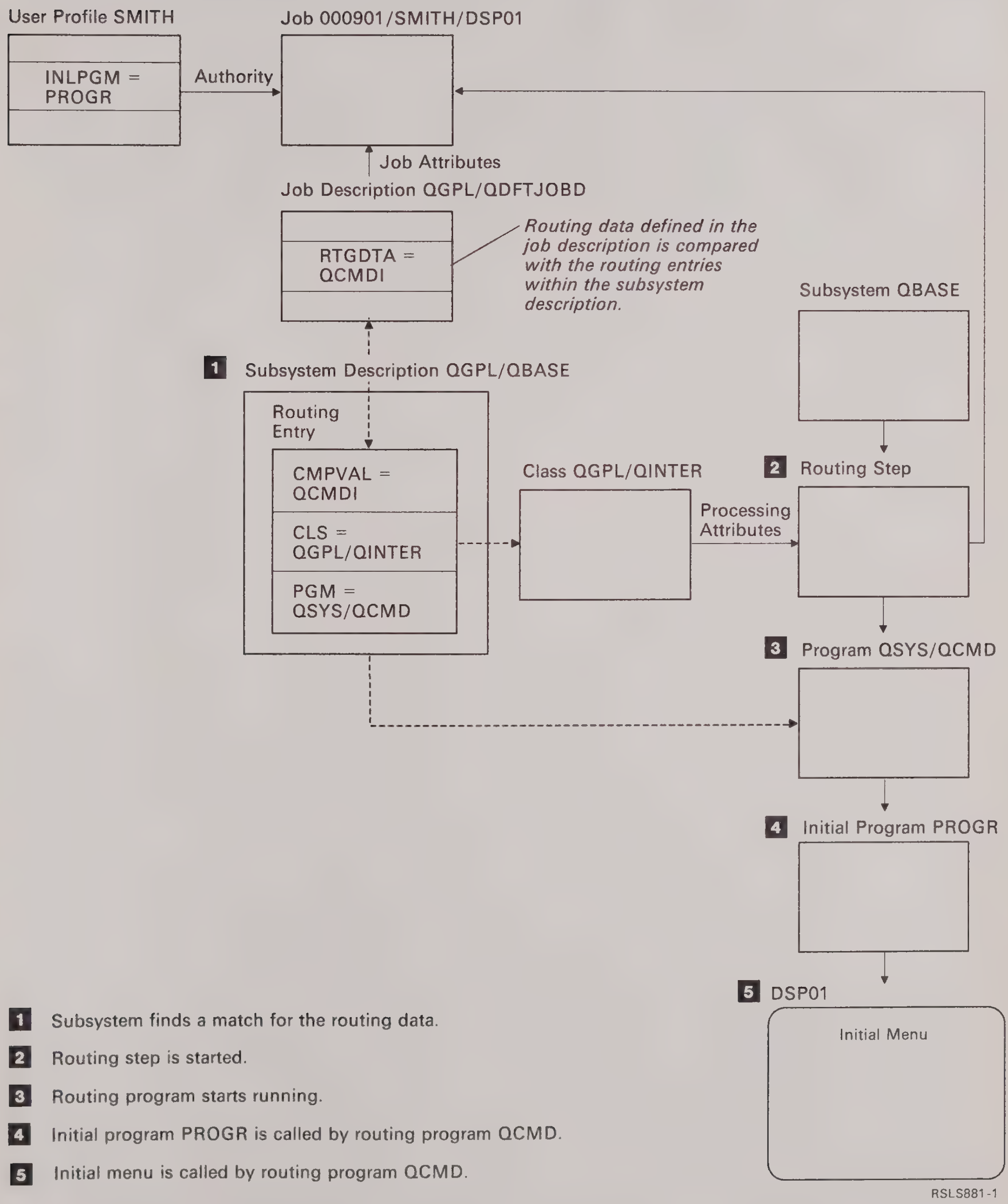
```
WRKJOB JOB(QPGMR/DSP01)
```

or

```
WRKJOB JOB(DSP01)
```

This works the same as entering the entire qualified job name as long as there is only one job on the system that matches the portion of the qualified name that you entered. If there is more than one job that matches, you will get a series of messages listing all the jobs that matched. Then you can enter the command again, specifying the fully qualified job name.

Figure 1-7 shows the subsequent activity leading up to starting a routing step and displaying the Main Menu for a user profile specifying an initial program.


User Profile SMITH
INLPGM = PROGR
Authority
Job 000901/SMITH/DSP01
Job Attributes
Job Description QGPL/QDFTJOBD
RTGDTA = QCMDI
Routing data defined in the job description is compared with the routing entries within the subsystem description.
Subsystem QBASE
1 Subsystem Description QGPL/QBASE
Routing Entry
CMPVAL = QCMDI
CLS = QGPL/QINTER
PGM = QSYS/QCMD
Class QGPL/QINTER
Processing Attributes
2 Routing Step
3 Program QSYS/QCMD
4 Initial Program PROGR
5 DSP01
Initial Menu
RSLS881-1

1. Subsystem finds a match for the routing data.
2. Routing step is started.
3. Routing program starts running.
4. Initial program PROGR is called by routing program QCMD.
5. Initial menu is called by routing program QCMD.

Figure 1-7. Subsystem Activity Leading Up to the Starting of a Routing Step and the Main Menu Display

The routing data is compared with the routing entries in the subsystem description. When a match is made, the program specified in the routing entry is called and the routing step is started. For the subsystem QBASE, the program QSYS/QCMD is called to process the commands in the job. QSYS/QCMD supports both interactive and batch jobs.

It should be apparent that user profiles play a large part in determining what a user sees at a work station and what a user can do on the system. Figure 1-8 on page 1-17 is a summary, organized by user profile, of the effect of shipped objects on running the system. Note the general description in the bottom portion of the figure. It illustrates a technique used in other shipped user profiles, which is one that you can use to tailor the AS/400 system to your needs.

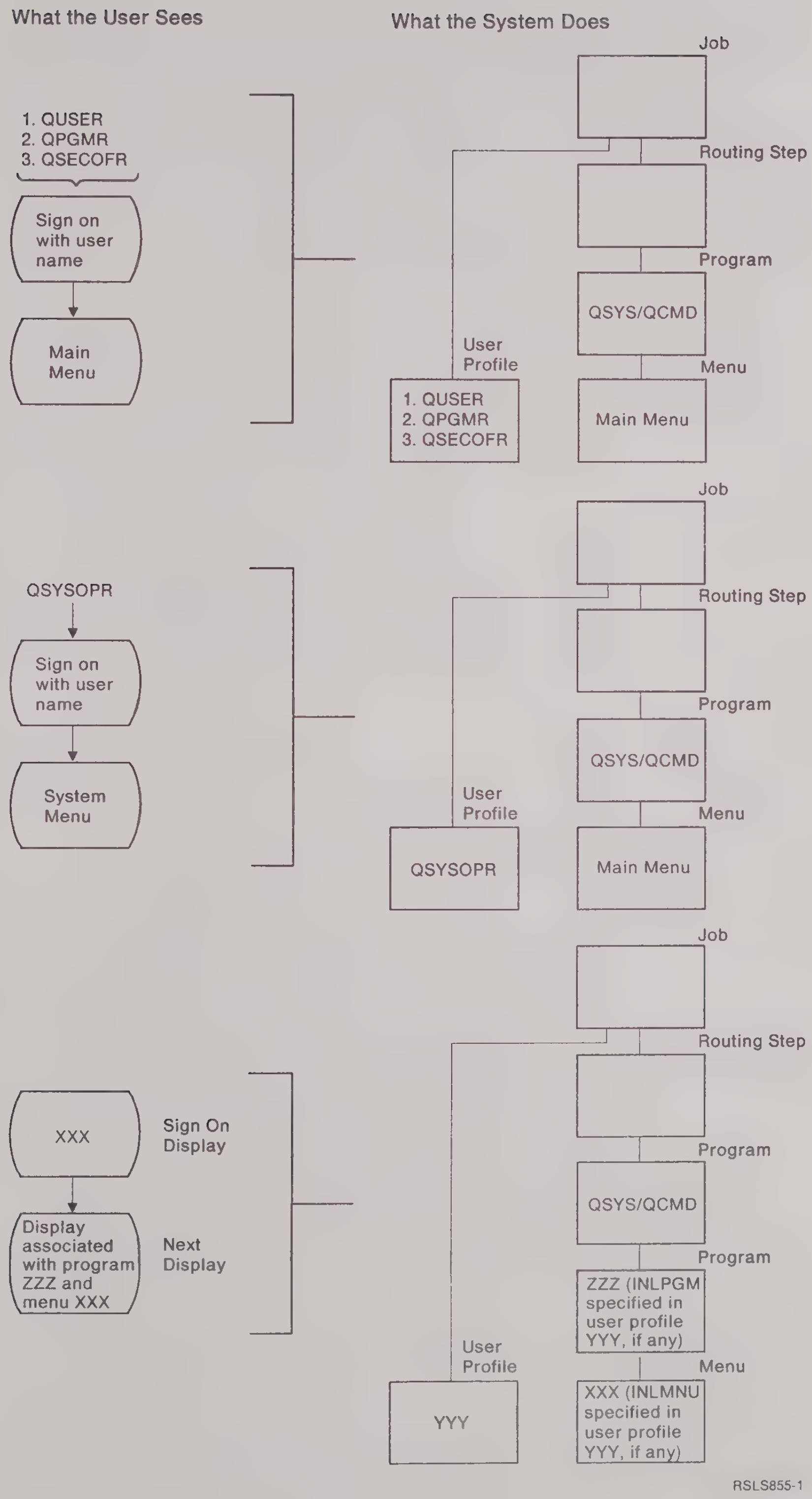

What the User Sees
What the System Does
1. QUSER
2. QPGMR
3. QSECOFR
Sign on with user name
Main Menu
Job
Routing Step
Program
QSYS/QCMD
User Profile
1. QUSER
2. QPGMR
3. QSECOFR
Menu
Main Menu
QSYSOPR
Sign on with user name
System Menu
Job
Routing Step
Program
QSYS/QCMD
User Profile
QSYSOPR
Menu
Main Menu
XXX
Sign On Display
Display associated with program ZZZ and menu XXX
Next Display
Job
Routing Step
Program
QSYS/QCMD
Program
ZZZ (INLPGM specified in user profile YYY, if any)
User Profile
YYY
Menu
XXX (INLMNU specified in user profile YYY, if any)
RSLS855-1

Figure 1-8. Summary of the Effect of the Shipped Objects on a Running AS/400 System

Use of Shipped Objects for Batch Jobs

The primary difference between the interactive support on the AS/400 system and the support for batch jobs is that the batch functions are processed as a result of entries placed on a job queue instead of signing on to a work station.

Figure 1-9 shows how the subsystem QBASE, which is started using the IBM-supplied subsystem description QGPL/QBASE, operates for batch jobs.

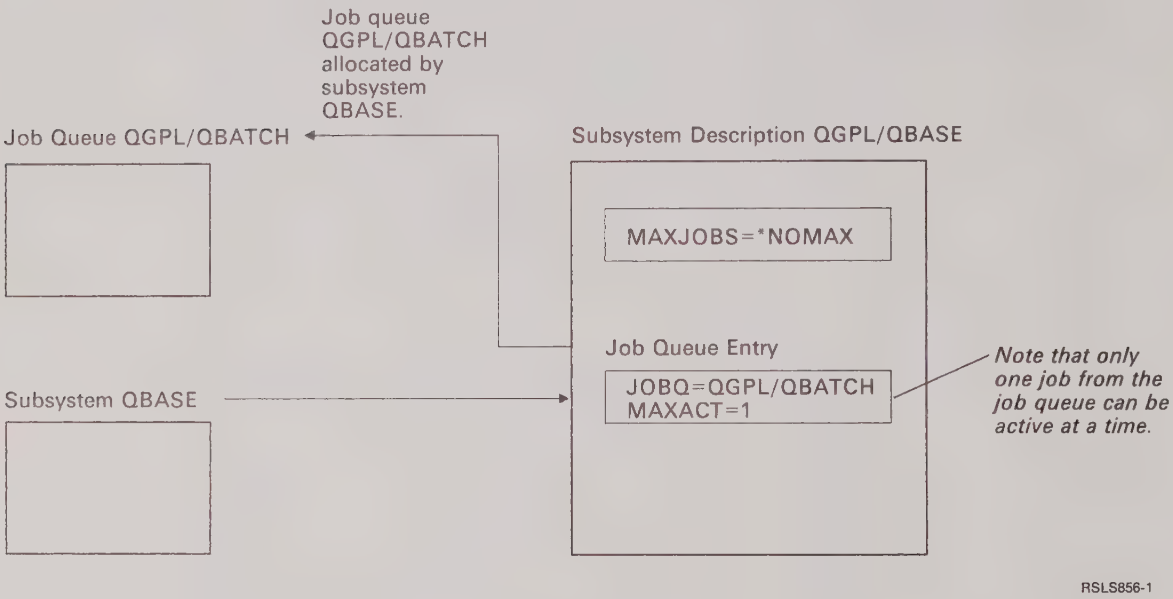

Job queue QGPL/QBATCH allocated by subsystem QBASE.
Job Queue QGPL/QBATCH
Subsystem Description QGPL/QBASE
MAXJOBS=*NOMAX
Job Queue Entry
JOBQ=QGPL/QBATCH
MAXACT=1
Note that only one job from the job queue can be active at a time.
Subsystem QBASE
RSLS856-1

Figure 1-9. How the Job Queue Is Identified to the QBASE Subsystem

A job queue entry in the subsystem description QGPL/QBASE specifies that jobs can be started using the job queue QGPL/QBATCH. Jobs can be placed on a job queue even if the subsystem has not been started. When the subsystem QBASE (or QBATCH, if using QCTL as the controlling subsystem) is started, it processes the jobs on the queue. A subsystem description can specify the maximum number of jobs (batch or interactive) that can be processed at the same time. For the job queue entry QBATCH in the subsystem QGPL/QBASE, only one batch job can be processed at any one time. (You can use the Change Job Queue Entry (CHGJOBQE) command to change this at any time.) The number of jobs that can be active from any job queue can be specified in the job queue entry.

Note that not all jobs on a job queue are necessarily available for processing when the subsystem is started; jobs can be held on a queue until the system operator releases them. If the subsystem is ended before all the jobs are processed, the jobs remain on the queue until the subsystem is started again or until another subsystem allocates the same job queue. More than one subsystem description can refer to the same job queue, but only one active subsystem at a time can use the job queue as a source of batch jobs. Therefore, if a subsystem ends and there are still jobs on the job queue, another subsystem referring to that job queue can be started to process the jobs. If another subsystem is already started and is waiting for the same job queue, the subsystem automatically allocates the job queue when it becomes available.

Routing steps for batch jobs are started in the same manner as for interactive jobs. That is, the routing data is compared with the routing entries in the subsystem description. When a match is made, the program specified in the routing entry is called and the routing step is started. For the subsystem QBASE, the program QSYS/QCMD is called to process the commands in the job. QSYS/QCMD supports both interactive and batch jobs.

The routing data is taken from the job description specified on the BCHJOB command or SBMJOB command that placed the job on the job queue, or from a RRTJOB, TFRJOB, or TFRBCHJOB command for a job already running. If routing data is specified in the BCHJOB or SBMJOB command, it is used instead of the routing data contained in the job description specified.

Jobs can be placed on the job queue QGPL/QBATCH (or any job queue) in more than one way:

- The system operator can issue a Start Data Base Reader (STRDBRDR) or Start Diskette Reader (STRDKTRDR) command to read input from a device or data base file. For each job in the input stream, an entry is placed on a job queue. For example, the STRDBRDR command starts a reader to read records from a data base member and places the job on the job queue QGPL/QBATCH. Jobs are delimited by BCHJOB and ENDBCHJOB commands.
- A SBMJOB command can be used to submit a job to a user-specified job queue. The default of the command is to submit the job to run with the current user's user profile and some of the same job attributes. All other defaults are taken from a named job description or the currently active job (the job issuing the SBMJOB command). If no job description is specified, the job description specified in your user profile is used.
- A SBMDBJOB command or SBMDKTJOB command can be used. For each job in the input stream, an entry is placed on a job queue. For example, the SBMDKTJOB command submits jobs from diskette to the job queue QGPL/QBATCH, which is the default. Jobs are delimited by BCHJOB and ENDBCHJOB commands.
- Active jobs can be placed on the queue by specifying the TFRJOB or the TFRBCHJOB command in the active job.

A batch job on any job queue can be moved to a different queue. For example, to move a job to the QBATCH job queue, type:

```
CHGJOB  JOB(xxxxx)  JOBQ(QBATCH)
```

When a job queue name is not specified on a start reader (STRXXXRDR) command or a submit jobs (SBMXXXJOB) command, the job is placed on the job queue

QGPL/QBATCH. When a job queue name is not specified on a SBMJOB command, the job is placed on the job queue named in the job description specified by the SBMJOB command. Jobs submitted with the SBMDBJOB, SBMDKTJOB, or SBMJOB command can be displayed by using the Work with Submitted Job (WRKSBMJOB) command, unless DSPSBMJOB(*NO) was specified when the job was submitted.

When you submit a batch job, you can specify attributes of the job in one of two ways:

- Use a specified job description without overriding any of the attributes.
- Use a specified job description but override some of the attributes (using the BCHJOB or SBMJOB command) for the job.

The job description QGPL/QBATCH is the default for the BCHJOB command. The default user profile for the BCHJOB command is QPGMR because it is specified in the job description QGPL/QBATCH. The default user profile for the SBMJOB command is *CURRENT; the submitted job uses the same profile as the job that submitted it.

Use of Shipped Objects for Spooling Jobs

Spooling is supported in a manner similar to batch jobs. Figure 1-10 shows how the spooling subsystem QSPL, which is started from the shipped subsystem description QGPL/QSPL, operates. QSPL supports the processing of spooling readers and writers, that transfer data from and to devices independently of the application processing.

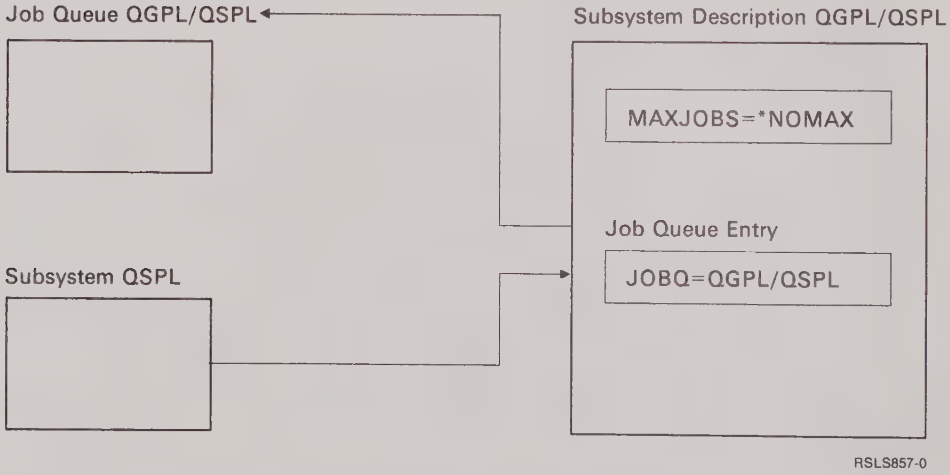


Figure 1-10. How the Job Queue Is Identified to the Spooling Subsystem

When the subsystem QSPL is started, it processes the jobs on the queue QGPL/QSPL until it is empty or the subsystem is ended.

Because a separate job description is used for each type of reader or writer, you can set up your system to uniquely handle different types of spooling processing. See the section on IBM-supplied objects in the *System Reference Summary*, for a description of the spooling job descriptions.

Use of Shipped Objects for Communications Device Allocation

When subsystems start, they request allocation of all communications devices in the communications entries in the subsystem description. The requests are sent to the QLUS (LU services) system job which handles device allocation for all communications devices.

QLUS gets notified when a communications device is available for program start request processing. This notification occurs when the connection between the local and remote system is established for that device. When QLUS receives this notification, it attempts to allocate the communications device to a subsystem based on communications entry definitions. If there is no subsystem active that wants to use the device, QLUS maintains allocation of the device until the device is varied off, or a subsystem starts that wants to use the device.

Rules for Device Allocation

When more than one communications entry exists for a communications device, QLUS uses the following rules to determine which subsystem will use the device (when the device is available):

- Communications entries with the highest level of detail for the device will be processed first. The order (from highest to lowest) of detail is: device name entry, remote location name entry, device type entry.
- Mode names are used only for advanced program-to-program communications(APPC)/advanced peer-to-peer networking (APPN) devices. Each mode on each device is allocated to a subsystem. A specific mode name will take priority over the generic *ANY mode name.
- The time that the subsystem requested the device (when the subsystem is started) is used to break ties when two or more subsystems have the same level of detail for the device and mode.

Once a communications device is allocated to a subsystem it remains in that subsystem until the subsystem deallocates the device. When a subsystem deallocates a device (and deallocation is not due to a device error or varying off the device), QLUS attempts to allocate the device to another subsystem.

If a subsystem has a communications device allocated and you start a second subsystem that should really have the device (based on the device allocation rules), you can force the original subsystem to deallocate the device. This deallocation causes QLUS to go through the device allocation algorithm again, which will eventually cause the device to be allocated by the second subsystem. Here are some ways to cause a subsystem to deallocate a communications device:

- Varying the device off and then on again causes QLUS to attempt to allocate the device to a subsystem.
- Issuing the Allocate Object (ALCOBJ) command against the device works for some communications types (request a *EXCLRD lock). Issuing the Deallocate Object (DLCOBJ) command causes QLUS to attempt to allocate the device to a subsystem.
- Ending the subsystem (ENDSBS command) causes QLUS to automatically attempt to allocate the device to a subsystem.

Subsystem Summary

The objects shipped with the system give you a considerable amount of system function with a minimum amount of effort on your part. The default subsystems, QBASE and QCTL, will automatically start spool printing and have all work stations ready to sign on. The system will also adjust storage pools and activity levels for you.

Once you understand how work management operates using the shipped objects and are familiar with the control language (CL) commands, you should be able to start using work management functions to tailor the system to your operating needs. You can continue to run with the shipped system as is or you can do the following:

- Change the contents of the shipped objects to adjust the basic system operation to your installation's needs. For example, you can change the size of storage pools to handle your application needs.
- Create your own subsystem descriptions, job descriptions, and classes to tailor the system to your installation's needs. Chapter 2, "Using Work Management Functions," gives you the information you need to do this.

Chapter 2. Using Work Management Functions

This chapter describes the work management objects (subsystem descriptions, job descriptions, and classes) and how they operate. It also describes what objects are shipped with the system. Examples and considerations of job starting and routing, and examples of creating your own subsystems are also described. It also lists the display and work-with commands that you can use to evaluate your system's performance.

Figure 2-1 on page 2-2 shows the work management objects, the commands to create them, and the parameters relevant to work management.

Subsystem Description

CRTSBSD
- POOLS
- MAXJOBS
- SGNDSPF

Work Station Entry

ADDWSE
- JOBD
- MAXACT
- AT

Routing Entry

ADDRTGE
- SEQNBR
- CMPVAL
- PGM
- CLS
- MAXACT
- POOLID

Job Queue Entry

ADDJOBQE
- JOBQ
- MAXACT
- SEQNBR
- MAXPTYn (n = 1-9)

Communications Entry

ADDCMNE
- JOBD
- DFTUSR
- MAXACT
- RMTLOCNAME
- DEV
- MODE

Autostart Job Entry

ADDAJE
- JOB
- JOBD

Job Description

CRTJOBD
- USER
- JOBQ
- JOBPTY
- OUTPTY
- RTGDTA
- RQSDTA
- PRTTXT
- ACGCDE
- PRTDEV
- OUTQ
- SYNTAX
- INLLIBL
- ENDSEV
- LOGCLPGM
- LOG
- INQMSGRPY
- HOLD
- DATE
- SWS
- DATFMT
- DATSEP

Class

CRTCLS
- RUNPTY
- TIMESLICE
- PURGE
- DFTWAIT
- CPUTIME
- MAXTMPSTG

User Profile

CRTUSRPRF
- SPCAUT
- SPCENV
- MAXSTG
- PTYLMT
- CURLIB
- INLPGM
- INLMNU
- JOBD
- ACGCDE
- MSGQ
- DLVRY
- SEV
- PRTDEV
- OUTQ

Job Queue

CRTJOBQ

Output Queue

CRTOUTQ

RSLS895-0

Figure 2-1. Summary of Work Management Objects, Commands, and Parameters

Work Management Objects

This section describes how you create subsystem descriptions, job descriptions, classes, job queues, and output queues.

Subsystem Descriptions

A subsystem description consists of three parts:

- Subsystem attributes (overall subsystem characteristics)
  - Storage pool definitions (identifiers, sizes, and activity levels)
  - Maximum number of jobs that can be active in a subsystem at the same time
  - Sign-on display file
  - Authority
  - Text description
- Work entries (sources of work)
  - Autostart job entry
  - Work station entry
  - Job queue entry
  - Communications entry
- Routing entries

You specify the subsystem attributes using the Create Subsystem Description (CRTSBSD) command. You add work entries to the subsystem description using the following commands:

- Add Autostart Job Entry (ADDAJE)
- Add Work Station Entry (ADDWSE)
- Add Job Queue Entry (ADDJOBQE)
- Add Communications Entry (ADDCMNE)

You can add routing entries to the subsystem description using the Add Routing Entry (ADDRTGE) command. You can change the IBM-supplied subsystem descriptions or any user-created subsystem descriptions by using the Change Subsystem Description (CHGSBSD) command. To change or remove a specific type of work entry, or to change or remove routing entries, you use the following commands:

- Change Autostart Job Entry (CHGAJE)
- Remove Autostart Job Entry (RMVAJE)
- Change Work Station Entry (CHGWSE)
- Remove Work Station Entry (RMVWSE)
- Change Job Queue Entry (CHGJOBQE)
- Remove Job Queue Entry (RMVJOBQE)
- Change Communications Entry (CHGCMNE)
- Remove Communications Entry (RMVCMNE)
- Change Routing Entry (CHGRTGE)
- Remove Routing Entry (RMVRTGE)

You can use the CHGSBSD command to change the storage pool size, storage pool activity level, sign-on display file, and the maximum number of jobs for the subsystem description of an active subsystem. You can also use the CHGJOBQE command to change an existing job queue entry while the subsystem is active. All other changes to the subsystem description can only be made when the subsystem is inactive, including adding or removing storage pools. (An active subsystem is one that has been started, for example, with the Start Subsystem (STRSBS) command; an inactive subsystem is one that has been ended, for example, with the End Subsystem (ENDSBS) command, or has not been started.)

You can use the Delete Subsystem Description (DLTSBSD) command to delete an inactive subsystem description, and you can use the Display Subsystem Description (DSPSBSD) command to display the current characteristics of a subsystem description.

Double-Byte Character Set Considerations: IBM-supplied subsystem descriptions are changed for controlling double-byte character set devices. You do not need to change any subsystem descriptions.

Storage Pool Definitions

On the AS/400 system, all main storage can be divided into logical allocations called storage pools.

When you receive your system, four storage pools are normally defined:

- A *machine storage pool* that is used for highly shared machine and OS/400 programs. The size for this storage pool is specified in the system value QMCHPOOL. No user jobs run in this storage pool. (On the system status display (WRKSYSSTS), the machine storage pool appears as system pool identifier 1.)
- A *base storage pool* that contains all unassigned main storage on the system; that is, all storage that is not required by the machine storage pool or by another active subsystem. The system value QBASPOOL specifies the minimum size of the base storage pool. The activity level for this storage pool is specified in the system value QBASACTLVL. This storage pool can be shared between subsystems. The base storage pool is used for batch work and miscellaneous system functions.

  The base storage pool always appears as pool 2 on the system status display (WRKSYSSTS).
- A storage pool that runs print writers in the shipped QSPL subsystem. (Readers and diskette writers route into *BASE pool.)
- A storage pool that runs interactive jobs.

  Note: This storage pool is defined if the initial program load (IPL) performance adjustment is set on.

When a user subsystem is defined, one or more pools may be defined, which are labeled 1, 2, 3, and so on. These are the designations of the subsystem pools, and do not correspond to the pool numbers shown on the system status display. The subsystems display (WRKSBS) relates the subsystem pool identifiers and the column headings to the system pool identifiers. The active jobs display (WRKACTJOB) shows the system pool identifier associated with the job.

There is a special type of job, called a subsystem monitor job, that controls all the activity in a subsystem. It is the first job started when the subsystem starts, and the last to end when the subsystem is ending. Its job name is the same as the name of the subsystem description used to start it. It uses the information in the subsystem description to control things like which work stations should have sign-on screens and how many batch jobs can be running at the same time. It also handles starting and ending user jobs. Storage for the subsystem monitor comes from pool 1 of the subsystem, but does not count toward an activity level.

The shipped default causes the storage for the subsystem monitor to come from the base storage pool.

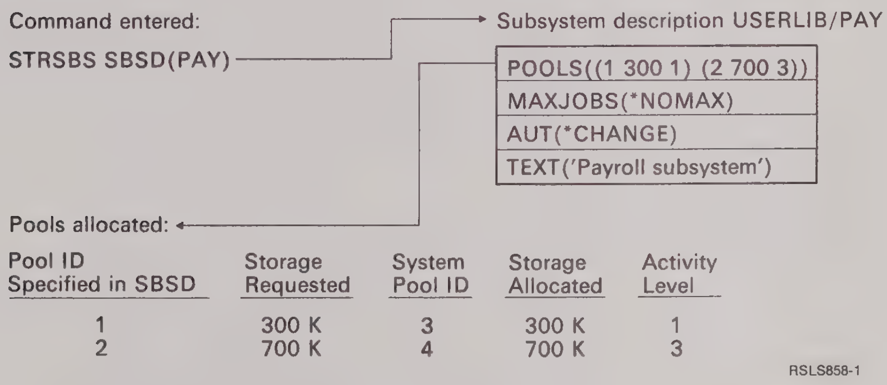
If you want storage for the subsystem monitor to come from a separate pool, specify the CRTSBSD or CHGSBSD command with a specific pool size and activity level such as:

```
POOLS((1 300 2))
```

For more information on the machine storage pool and the base storage pool, see Chapter 5, "Performance Guidelines."

In addition to the machine storage pool and the base storage pool, you can create user-defined storage pools. User-defined storage pools can be used by IBM-supplied subsystems or user-defined subsystems. To create a user-defined storage pool, use the POOLS parameter on the Create Subsystem Description (CRTSBSD) or Change Subsystem Description (CHGSBSD) commands.

The following example shows how, when a subsystem is started, the system attempts to allocate the user-defined storage pools defined in the subsystem description of the started subsystem:



```
STRSBS SBSD(PAY)
```

```
POOLS((1 300 1) (2 700 3))
MAXJOBS(*NOMAX)
AUT(*CHANGE)
TEXT('Payroll subsystem')
```

Pool ID Specified in SBSD	Storage Requested	System Pool ID	Storage Allocated	Activity Level
1	300 K	3	300 K	1
2	700 K	4	700 K	3

If the system cannot allocate all the requested storage, it allocates as much storage as is available and allocates all the other as storage becomes available. For example, if, in the previous example, there was only 700K available, 300K would have been allocated to the first storage pool and 400K would have been allocated to the second storage pool.

The storage pools that you define decrease the size of the base storage pool. The system will not allocate storage to a user pool if the amount of storage available to the base storage pool would be less than the minimum base pool size specified by the system value QBASPOOL.

In the user-defined PAY subsystem jobs can be run in user-defined pool 1 or pool 2, as a result of the pool identifier specified in the associated routing entry (see "Routing Entries" on page 2-12 later in this chapter).

You can specify an activity level for a storage pool. This activity level specifies the maximum number of jobs that can run at the same time (compete for the use of the processing unit) in the storage pool. In subsystem PAY, the activity level in pool 1 is 1 and the activity level in pool 2 is 3. (See "Summary of Controls on Levels of Activity" on page 2-45 later in this chapter for a description of the relationship between the storage pool activity level and the MAXACT parameters for work entries and the MAXJOBS parameter for the subsystem description.) See Chapter 5, "Performance Guidelines" for a more detailed description of activity levels.

Once the maximum activity level for a storage pool has been reached, additional jobs needing the storage pool are automatically put in a queue (placed in an ineligible state, that is shown on the system status display). As soon as a job gives up its use of the storage pool, the jobs in the queue become eligible to run by their priority. For example, if a running job is waiting for a response from a work station, it gives up its activity level and the activity level is no longer at its maximum.

Defining storage pools and activity levels correctly is generally dependent on the number of jobs a subsystem must support at the same time and the characteristics and use of the programs that perform functions in those jobs.

The storage pool that user jobs get their storage from is always the same pool that limits their activity level. System jobs (such as SCPF, QSYSARB, and QLUS) get their storage from the base pool, but use the machine pool activity level. Subsystem monitors get their storage from the first subsystem description pool but not the activity level. This allows a subsystem monitor to always be able to run regardless of the activity level setting.

Work Entries

Work entries identify the sources for jobs that are processed in a subsystem. You add the work entries to the subsystem description with the Add Autostart Job Entry (ADDAJE), Add Work Station Entry (ADDWSE), Add Job Queue Entry (ADDJOBQE), or Add Communications Entry (ADDCMNE) command. There are four types of work entries:

- Autostart job entries. The job is automatically started each time the subsystem is started.
- Job queue entries. Jobs to be processed are taken from the specified job queue.
- Work station entries. The job is processed when a work station user signs on or when a work station user transfers an interactive job from another subsystem.
- Communications entries. The job is processed when the subsystem receives a communications program start request from a remote system.

You can specify the following in these work entries (that are part of the subsystem description). Parameter names are given in parentheses.

- For an autostart job entry:
  - Job name (JOB)
  - Job description name (JOBD)
- For a job queue entry:
  - Job queue name (JOBQ)
  - Maximum number of jobs that can be active at the same time from the queue (MAXACT)
  - Order in which the subsystem selects job queues from which jobs can be started (SEQNBR)
  - Maximum number of jobs that can be active at the same time for a specified job queue priority (MAXPTYN)
- For a work station entry:
  - Work station name or type (WRKSTN or WRKSTNTYPE)
  - Job description name (JOBD) or job description name in the user profile
  - Maximum number of jobs that can be active at the same time through the entry (MAXACT)
  - When the work stations are to be allocated, either when the subsystem is started or when an interactive job enters the subsystem through the Transfer Job (TFRJOB) command (AT)
- For a communications entry:
  - Device description name or type (DEV)
  - Remote location name (RMTLOCNAME)
  - Job description name (JOBD), or job description name in the user profile
  - Maximum number of jobs that can be active through this communications entry (MAXACT)
  - Default user (DFTUSR)
  - Mode name (MODE)

When you specify more than one job queue for a subsystem, the subsystem selects jobs from the job queue with the lowest sequence number (SEQNBR parameter on the ADDJOBQE command) until it reaches its maximum activity value or the job queue has no more jobs to be processed. The subsystem then selects jobs from the job queue with the next higher sequence number. This process continues until all job queue entries have been processed or until the subsystem reaches its maximum activity value (as specified by the MAXJOBS parameter in the subsystem description) as shown in the following example:

Job Queue A (SEQNBR = 10)
Job 1
Job 2
Job 3

Job Queue B (SEQNBR = 20)

Job 4
Job 5
Job 6

Job Queue C (SEQNBR = 30)

Job 7
Job 8
Job 9

Each job queue entry in this example is specified as MAXACT(*NOMAX). The subsystem first selects jobs from job queue A because the job queue entry has the lowest sequence number. If the subsystem's maximum activity level is 3 (MAXJOBS(3) parameter on the CRTSBSD command), it can select all the jobs from job queue A to be active at the same time. When any of the three jobs is completed, the activity level is no longer at the maximum; therefore a new job is selected from job queue B because it has the next lowest sequence number (assuming no new jobs have been added to job queue A).

Because each job queue entry specifies MAXACT(*NOMAX), an unlimited number of jobs can be started. Had each job queue entry specified MAXACT(1), then jobs 1, 4, and 7 would have been started. Had job queue entry A been specified as MAXACT(2), then jobs 1, 2, and 4 would have been started.

When a subsystem is started, the subsystem attempts to allocate all of its job queues. This allocation may fail if the job queue does not exist, cannot be locked, or is allocated to another subsystem.

If a job queue does not exist when the subsystem is started, the job queue is allocated to the subsystem when:

- The job queue is created.
- A job queue is renamed with the name defined to the subsystem.
- A job queue is moved to another library and the resulting qualified name matches the name in the subsystem description.
- The library containing the job queue is renamed and the resulting qualified name matches the name in the subsystem description.

When a subsystem is started, it attempts to allocate all devices in its subsystem description. The following situations may occur during the time the subsystem starts:

- If the device is not varied on, the subsystem cannot allocate it. The system arbiter (QSYSARB) holds a lock on all varied-off devices.
- If the device is varied on and has not been allocated by any other subsystem, the subsystem can allocate it and display the Sign On display.
- If the device is varied on and has been allocated by another subsystem and is at the Sign On display, (the Sign On display was displayed before the second subsystem was started), a second subsystem can allocate the device from the first subsystem and display the Sign On display.

If more than one subsystem tries to allocate the same work station (as specified in the work station entries) and the work station is varied off, which subsystem that gets the work station when it is varied on cannot be predicted. Similarly, if a work station entry specifies a work station type instead of a work station name, a subsystem may get all, some, or none of the work stations of that type. To avoid such a situation, you can do one of the following:

- Vary the work station on when the system is started by specifying ONLINE(*YES) in the device description for the work station. The order in which the subsystems start will still determine what subsystem gets the work station.
- Set up the work station entries for the subsystems so that multiple subsystems are not using the same work stations.

When a user signs on to a device, his job runs in the subsystem that had the Sign On display displayed on the device. The following situations may occur after the user has signed on:

- If a second subsystem is started and it tries to allocate the device on which the user signed on, the second subsystem cannot allocate it. The user's job continues to run in the first subsystem.
- If the user selects Display sign on for alternative job (option 1) on the System Request menu or issues the Transfer to Secondary Job (TFRSECJOB) command, the new job runs in the same subsystem as the original job.
- When the user signs off, the device remains allocated to the subsystem used when he signed on, and the Sign On display is displayed. Any subsequent jobs from that device will continue to run in that subsystem, (unless another subsystem is started up that allocates the device while it is at the Sign On display).
- If the user signs off and the subsystem in which his job was running is ended, the device is deallocated. A second subsystem can then allocate the device and display the Sign On display.

Example of Device Allocation

In this example a subsystem A and subsystem B have work stations DSP01 and DSP02 in their subsystem description (the work station entries specify AT(*SIGNON)).

Device Name	Allocated to:
DSP01	Subsystem A
DSP02	Subsystem A

Assume that both work stations are varied on when subsystem A is started. Subsystem A allocates both work stations and displays the Sign On display on both. Even though subsystem A has the Sign On display displayed on the work stations, they can be allocated by another subsystem or job; the work station would then no longer be available to subsystem A.

Device Name	Allocated to:
DSP01	USER1
DSP02	Subsystem A

When a user (USER1) signs on to work station DSP01, the device is allocated to USER1's job, which is running in subsystem A. Work station DSP02 is still at the Sign On display; thus it can be allocated by another subsystem or job; it would then no longer be available to subsystem A.

Device Name	Allocated to:
DSP01	USER1
DSP02	Subsystem B

Subsystem B is started. Because USER1 has signed on to work station DSP01, subsystem B cannot allocate the device. Subsystem B requests allocation of the device when it becomes available. DSP02 is allocated to subsystem B because no one had signed on to it in subsystem A. Any jobs started on DSP02 will run in subsystem B.

Device Name	Allocated to:
DSP01	Subsystem A
DSP02	Subsystem B

USER1 signs off. Because the user job was running in subsystem A, that subsystem displays the Sign On display so that another user can sign on the work station and run in subsystem A. If subsystem A is ended, work station DSP01 will be allocated by subsystem B (because it has an outstanding request to allocate the device.)

The name of the subsystem that currently has a work station allocated appears in the upper right corner of the Sign On display.

Changing the Sign-On Display File

If you want to change the format of the Sign On display, you can do the following:

1. Create a changed sign-on display file.
2. Change a subsystem description to use the changed display file instead of the system default of QSYS/QDSIGNON.
3. Test the change.
4. Change other subsystem descriptions.

Display File Source

The source for the sign-on display file is shipped as a member (QDSIGNON) in the QDDSSRC physical file.

How to Change the Display File

There is a hidden field in the display file named UBUFFER. It is 128 bytes long and is stated as the last field in the display file. This field can be changed to function as an input/output buffer so the data specified on this portion of the screen will be available to application programs when the interactive job is started.

You can change the UBUFFER field to contain as many smaller fields as you need if the following requirements are met:

1. The new fields must follow all other fields in the display file. The location of the fields on the screen does not matter as long as the order in which they are put in the data description specifications (DDS) meets this requirement.
2. The length must total 128. If the length of the fields is more than 128, some of the data will not be passed.
3. All fields must be input/output fields (type B in DDS source) or hidden fields (type H in DDS source).

Notes:

1. The order in which the fields in the sign-on display file are declared must not be changed. The position in which they are displayed on the screen can be changed.
2. Do not change the total size of the input or output buffers. Serious problems can occur if the order or size of the buffers are changed.

When creating the sign-on display file with the Create Display File (CRTDSPF) command, you should specify the maximum number of devices expected to be used by the subsystem on the MAXDEV parameter (up to 256; the default is 1). This improves performance when the subsystem is doing I/O to the sign-on display file. You can use more than 256 devices but you cannot specify more than 256 on this parameter. For the best performance, you should specify no less than the maximum number of devices you want sign-on screens on (up to the maximum of 256).

Changing the Subsystem Description

After the new sign-on display file has been created by using the CRTDSPF command, you can change the subsystem descriptions for subsystems that you want to use the new display by using the CHGSBSD command and specify the new display file on the SGNDSPF parameter. You should use a test version of a subsystem to verify that the display is valid before attempting to change the controlling subsystem.

Retrieving the Sign-On Information in an Application Program

The data specified on the user portion of the Sign On display will be sent as replacement data in the CPF1124 message for interactive jobs. This data starts in byte 133 of the replacement text. The following program will retrieve the data:

```
PGM
DCL VAR(&MSGDTA)  TYPE(*CHAR) LEN(260) /*RECEIVES THE MESSAGE DATA */
DCL VAR(&MSGID)   TYPE(*CHAR) LEN(7)   /*RECEIVES THE MESSAGE ID   */
DCL VAR(&USERDTA) TYPE(*CHAR) LEN(128) /*RECEIVES THE USER DATA    */
/* Receive the CPF1124 message to obtain the user portion of the   */
/* sign-on display. Note that the message is not removed so that   */
/* it will be available to the job log. The message ID is also     */
/* retrieved to ensure that the message is CPF1124.                */
RCVMSG    PGMQ(*EXT) RMV(*NO) MSGDTA(&MSGDTA) MSGID(&MSGID)
IF        COND(&MSGID *EQ 'CPF1124') THEN(CHGVAR VAR(&USERDTA) +
          VALUE(%SST(&MSGDTA 133 128)))
ENDPGM
```

Using the Retrieved Data

After the retrieve program has run, the data that the user specified will be located in bytes 133 through 260 in the variable &MSGDTA. The CHGVAR command in the sample program moves the data to &USERDTA. The portions of the 128-byte buffer that were not used will appear as blanks in the variable &USERDTA.

Routing Entries

A routing entry defines how a job is to be started. You use the Add Routing Entry (ADDRTGE) command to add a routing entry. A routing entry contains the following (parameter names are given in parentheses):

- A routing entry sequence number (SEQNBR)
- A routing data comparison value and starting position for the comparison (CMPVAL)
- The name of the program to be called (PGM) or *RTGDTA, which indicates that the program name is to be taken from the routing data
- The name of the class to be used for the routing step (CLS)
- The maximum number of routing steps that can be active at the same time for the entry (MAXACT)
- The identifier of the subsystem storage pool in which the job is to run (POOLID)

For communications considerations for jobs started by program start requests, see the *Communications User's Guide* and the *Communications Programmer's Guide.*

When you add a routing entry to a subsystem description, you assign a sequence number to the entry. This sequence number tells the subsystem the order in which routing entries are to be searched for a routing data match. For example, you have a subsystem description containing the following five routing entries:

Sequence Number	Comparison Value
10	'ABC'
20	'AB'
30	'A'
40	'E'
50	'D'

The routing entries are searched in sequence number order. If the routing data is 'A', the search ends with routing entry 30. If the routing data is 'AB', the search ends with routing entry 20. If the routing data is 'ABC', the search ends with routing entry 10. Because routing data can be longer than the comparison value of the routing entry, the comparison (which is done in left-to-right order) stops when it reaches the end of the comparison value. Therefore, if the routing data is 'ABCD', the search ends with routing entry 10.

When you define routing entries, they must be ordered from the most specific to the most general. The following example shows a correct and incorrect way to define routing entries:

CORRECT		INCORRECT	
Sequence Number	**Comparison Value**	**Sequence Number**	**Comparison Value**
10	'ABC'	10	'ABC'
20	'AB'	20	'ABCD'
30	'A'	30	
40	'E'	40	

In the incorrect example, it is no longer possible to match routing entry 20 because any routing data that matches the comparison value for routing entry 20 matches the routing entry 10 first. When a routing entry is changed or added to a subsystem description with a comparison value that causes this situation, the system sends a diagnostic message identifying the situation.

When you add routing entries to a subsystem description, you should order them so that the entries likely to be compared most often are first. This reduces the search time.

You can specify a comparison value of *ANY on the highest numbered routing entry. *ANY means that a match is forced regardless of the routing data. Only one routing entry can contain the comparison value of *ANY, and it must be the last (highest sequence number) entry in the subsystem description.

The program named in the routing entry is given control when the routing step for the job is started. Parameters to control the run-time environment (priority, time slice, and so on) of the routing step for the job are taken from the class specified in the routing entry. (See "Classes" on page 2-15.)

Job Descriptions

A job description is used to collect many of the parameters used when starting a job. The job description gets rid of the need to specify the same parameters repeatedly for each job. You create a job description by using the Create Job Description (CRTJOBD) command. The following example and table show how you can create one job description and use it so you do not have to specify several parameters in a lot of commands:

```
CRTJOBD   JOBD(BATCH4)  USER(*RQD)  JOBPTY(4)  OUTPTY(4)  RTGDTA(QCMDI)
```

Command	**Parameter**	**Value from JOBD on CRTJOBD Command Above**
ADDAJE	JOBD	BATCH4
ADDCMNE	JOBD	BATCH4
BCHJOB	JOBD	BATCH4
CRTUSRPRF	JOBD	BATCH4
SBMJOB	JOBD	BATCH4

The following list describes some of the parameters on the CRTJOBD command that can be specified that are important to work management:

- User name (USER). The default for the user profile the job is to use.
- Job queue (JOBQ). The job queue on which the job is to be placed (batch job only).
- Job priority (JOBPTY). Priority of the job on the job queue (batch job only).
- Output priority (OUTPTY). Priority of any spooled output files created by the job.
- Routing data (RTGDTA). Data used to determine which routing entry to use for the job.
- Request data (RQSDTA). Data to be put on the job message queue.
- Initial library list (INLLIBL). The user part of the initial library list for the job.
- Printer device (PRTDEV). The default printer device on which the spooled output files made by the job are to be printed.
- Output queue (OUTQ). The default output queue on which the spooled output files made by the job are to be placed. (Files are placed on this queue if OUTQ(*JOB) is specified for the file or in an override command.) This defaults to the output queue associated with the printer device.

When you define a job, you can use the job description in one of two ways:

- Use a specified job description without overriding any of its attributes. For example:

  ```
  //BCHJOB JOB(OEDAILY) JOBD(QBATCH)
  ```

- Use a specified job description but override some of the attributes (using the BCHJOB or SBMJOB command). For example, to override the end severity in the job description QBATCH, you specify:

  ```
  //BCHJOB JOB(OEDAILY) JOBD(QBATCH) ENDSEV(35)
  ```

You cannot override any job description attributes for autostart jobs, work station jobs, or communications jobs.

The following commands can also be used with job descriptions:

- Change Job Description (CHGJOBD): Changes the contents of a job description.
- Delete Job Description (DLTJOBD): Deletes a job description.
- Display Job Description (DSPJOBD): Displays the contents of a job description.

Job Description Security Considerations

A job description that has a user profile name (USER) specified should be authorized only to specific individuals. If you create a job description with a user name (USER) other than *REQUIRED, the job description can be used by other users (other than the user named in the job description USER parameter) to run jobs. For example, if you specified:

```
CRTJOBD JOBD(XX) USER(JONES) . . . AUT(*USE)
```

In this example, any user can submit a job using the XX job description, and be authorized to whatever JONES is authorized to. To avoid any security exposure, **do not** authorize this type of job description to *PUBLIC. Also, if this type of job description is used on a work station entry, it allows anyone to sign on as that user just by pressing the Enter key.

Classes

You can create a class using the Create Class (CRTCLS) command. It contains parameters to control the run-time environment of a job. Some of the parameters that are important to work management are:

- Machine run priority (RUNPTY). The priority (1 through 99, with 1 as the highest priority) of a job when competing with other jobs for machine resources.
- Purge (PURGE). Indicates if the working storage for a routing step is to be removed from main storage and placed in auxiliary storage at the end of a time slice or when entering a long wait.
- Time slice (TIMESLICE). The quantity of processor time (specified in milliseconds) allowed for a job before other waiting jobs are given the opportunity to run.
- Default maximum wait time (DFTWAIT). Specifies the default maximum wait time (in seconds) that processing of a job is to be held up until the AS/400 instruction that performs a wait completes its running, such as a lock instruction.
- Maximum processing time (CPUTIME). The maximum amount of processor time (specified in milliseconds) a job can run before it is automatically ended by the system.
- Maximum temporary storage (MAXTMPSTG). The maximum amount of temporary auxiliary storage (specified in K bytes) a job can use before it is ended by the system.

Note: Maximum processing unit time and maximum temporary storage limits can help prevent an erroneous program from impairing system performance. However, a sufficient amount of processing time and temporary storage must be given to allow the job to complete its function.

The IBM-supplied classes are designed to meet the needs of typical interactive and batch applications. You can create your own classes. For example, you can create new classes to give you a wider range of machine running priorities.

The priorities for jobs that run in batch environments should normally be lower than priorities for jobs in interactive environments. You may want the priority for the system operator's jobs to be higher than priorities of other jobs so that the system operator can effectively respond to system needs. Also, the time slice should be small enough so that a looping program does not dominate processor time and an activity level.

If you use QCTL as the controlling subsystem, the operator will automatically be running at a higher priority after signing on at the console. This is because QCTL routes the console job using the QCTL class, which specifies a higher priority. Another way you could set up your system so that the operator would run at a higher priority would be to do the following:

1. When the subsystem that the operator would be running in is not active, add a routing entry to it with unique routing data and specify the QSYS/QCTL class.
2. Create a new job description for the operator, specifying the same unique routing data that you used on the routing entry.
3. Change the operator's user profile to specify the new job description.
4. Start the subsystem.
5. Now when the operator signs on to that subsystem, the job will route using the QCTL class, which specifies a higher priority than the class used by normal interactive jobs.

You can use the Delete Class (DLTCLS) command to delete a class, and you can use the Display Class (DSPCLS) command to display the contents of a class. If you want to change a class, you use the DLTCLS command and then the CRTCLS command.

You can use the Change Job (CHGJOB) command to change the following class attributes for a job's routing step if you have special job control authority (*JOBCTL authority specified in your user profile):

- Machine run priority (RUNPTY)
- Time slice (TIMESLICE)
- Purge (PURGE)
- Default maximum instruction wait time (DFTWAIT)

If you change any of these attributes, the changes are in effect only for the duration of the job. If you change the attributes for a job that is on a job queue, the changes take effect when the job becomes active.

Job and Output Queues for Spooling

When work (a job) enters the system, it is placed in a waiting line called a job queue. Jobs (work) on the job queue are then selected to run. Running jobs that are to produce printed output place entries on an output queue (waiting line) to be printed. This process is called spooling.

Spooling support is available for two types of queues:

- Job queues. Entries for jobs are placed on job queues by readers, by the Submit Data Base Jobs (SBMDBJOB) command, by the Submit Diskette Jobs (SBMDKTJOB) command, by the Submit Job (SBMJOB) command, or by transferring to a subsystem using the Transfer Job (TFRJOB) or Transfer Batch Job (TFRBCHJOB) command. A subsystem selects jobs from a job queue for running.
- Output queues. Entries for the spooled output files made by jobs are placed on an output queue. A writer selects the output from the queue for writing to an output device.

Both the job queue name and the default output queue name for a job are specified in the job description for the job. Both can be overridden using parameters on the BCHJOB and SBMJOB commands. In addition, the output queue specified for a job already on a job queue or active in the system can be changed for files not yet opened by the job by using the Change Job (CHGJOB) command. Job queue entries in a subsystem description specify which job queues a subsystem is to receive jobs from.

Submit Commands

The SBMDBJOB and SBMDKTJOB commands perform the same functions as readers in that they interpret input streams from diskette(s) or a data base file and place jobs on job queues.

The SBMDBJOB and SBMDKTJOB commands:

- Run in the same job as the requester of the command. (The work station is locked until the submit jobs command has completed running.)
- Do not perform any syntax checking of the jobs in the input stream.
- Require fewer resources than a reader because a separate job is not started.
- Process the input stream sooner than a reader.

Whereas a reader:

- Runs in its own job. (Once the command has been specified, you can continue with other work.)
- Performs syntax checking on any job in the input stream in which syntax checking is specified on the BCHJOB command or in the job description.
- Requires more system resources than the submit jobs commands because a separate job is created.

In general, you may prefer to use the SBMXXXJOB commands unless the input stream is large or unless you want to use the syntax checking function done by a reader.

The Submit Job (SBMJOB) command and the submit jobs (SBMDBJOB and SBMDKTJOB) commands all can be used to submit batch jobs to a batch job queue. The difference in these commands is where the job comes from:

- The SBMJOB command can be used to submit a job to a batch job queue by specifying a job description and by specifying a CL command or request data, or specifying routing data to run a program.
- If you want to run a single CL command in the batch job, the simplest way is to use the CMD parameter on SBMJOB, which does syntax checking and allows prompting.
- The SBMDBJOB and SBMDKTJOB commands can be used to submit a job to a batch job queue from a diskette device, or a data base file. For these jobs, the job description comes from the BCHJOB statement in the input stream.

For a detailed description of these commands, see the *CL Reference*.

In the following example, the SBMJOB command submits a job named WSYS, using the job description QBATCH, to the job queue QBATCH. The CMD parameter gives the CL command that will run in the job.

```
SBMJOB  JOBD(QBATCH)  JOB(WSYS)  JOBQ(QBATCH)  CMD(WRKSYS)
```

Job Queues

You can create job queues for groups of jobs such as low priority jobs and night jobs. The parameters on the Create Job Queue (CRTJOBQ) command specify (parameter names are given in parentheses):

- If a user having job control special authority can control the job queue and its contents (OPRCTL) (for example, if the job can be ended by someone other than the user who submitted the job)
- Authority (AUT)
- Text description (TEXT)

Output Queues

You can create output queues for particular uses such as by department and form types. The system automatically creates an output queue when you create a printer device (the same name is used). The parameters on the Create Output Queue (CRTOUTQ) command specify (parameter names are given in parentheses):

- If users that have authority to read the output queue can display or copy output data from files other than their own (DSPDTA)
- How many, if any, job separators (pages) are to be made between each job's output when written to the output device (does not apply to diskette) (JOBSEP)
- If a user having job control authority can control the output queue and its contents (OPRCTL)
- If files should be ordered on the queue by job start time or spooled file creation time (SEQ)
- Authority (AUT)
- Text description (TEXT)

For a list of all the commands that can be used for input/output spooling, see the command grouping lists in the *CL Reference*.

Summary of Work Management Objects

The following shows what work management information is specified in the various work management objects. Command names and parameter names are used to identify the information and where it is specified. Only the parameter names relevant to work management are shown.

Subsystem Description

POOLS	MAXJOBS	SGNDSPF	CRTSBSD
JOB	JOBD		ADDAJE
JOBQ	MAXACT	SEQNBR	ADDJOBQE
MAXPTYN			
WRKSTN or WRKSTNTYPE			ADDWSE
JOBD	MAXACT		
AT			
SEQNBR	CMPVAL	PGM	ADDRTGE
CLS	MAXACT	POOLID	
DEV	JOBD	MODE	ADDCMNE
DFTUSR	MAXACT		
RMTLOCNAME			

Job Description

USER	JOBQ	JOBPTY	CRTJOBD
OUTPTY	RTGDTA	RQSDTA	
PRTTXT	ACGCDE	PRTDEV	
OUTQ	SYNTAX	INLLIBL	
ENDSEV	LOGCLPGM	LOG	
INQMSGRPY	HOLD	DATE	
SWS	DATFMT	DATSEP	

Class

RUNPTY	TIMESLICE	CRTCLS
PURGE	DFTWAIT	
CPUTIME	MAXTMPSTG	

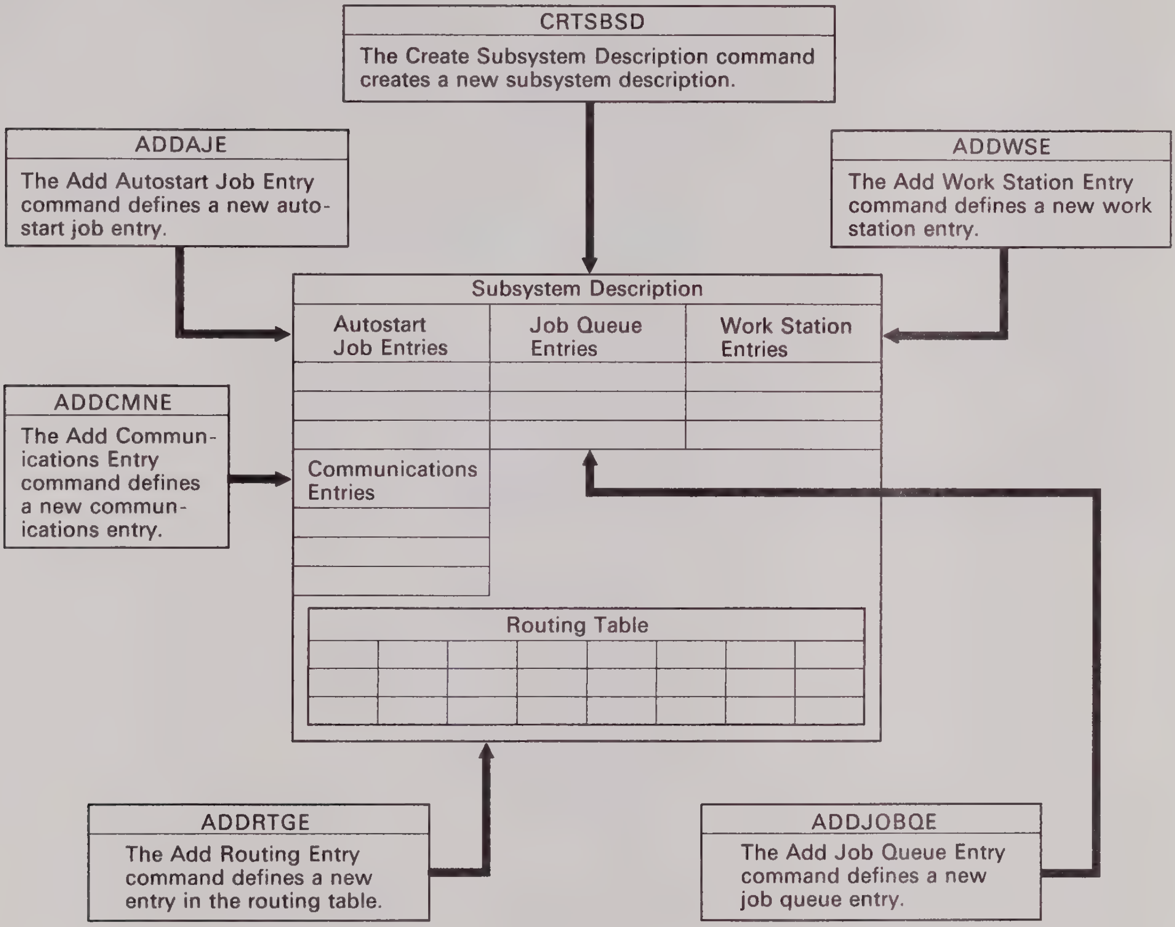
User Profile

SPCAUT	SPCENV	MAXSTG	CRTUSRPRF
PTYLMT	CURLIB	INLPGM	
INLMNU	JOBD	ACGCDE	
MSGQ	DLVRY	SEV	
PRTDEV	OUTQ		

Job Queue	CRTJOBQ
Output Queue	CRTOUTQ

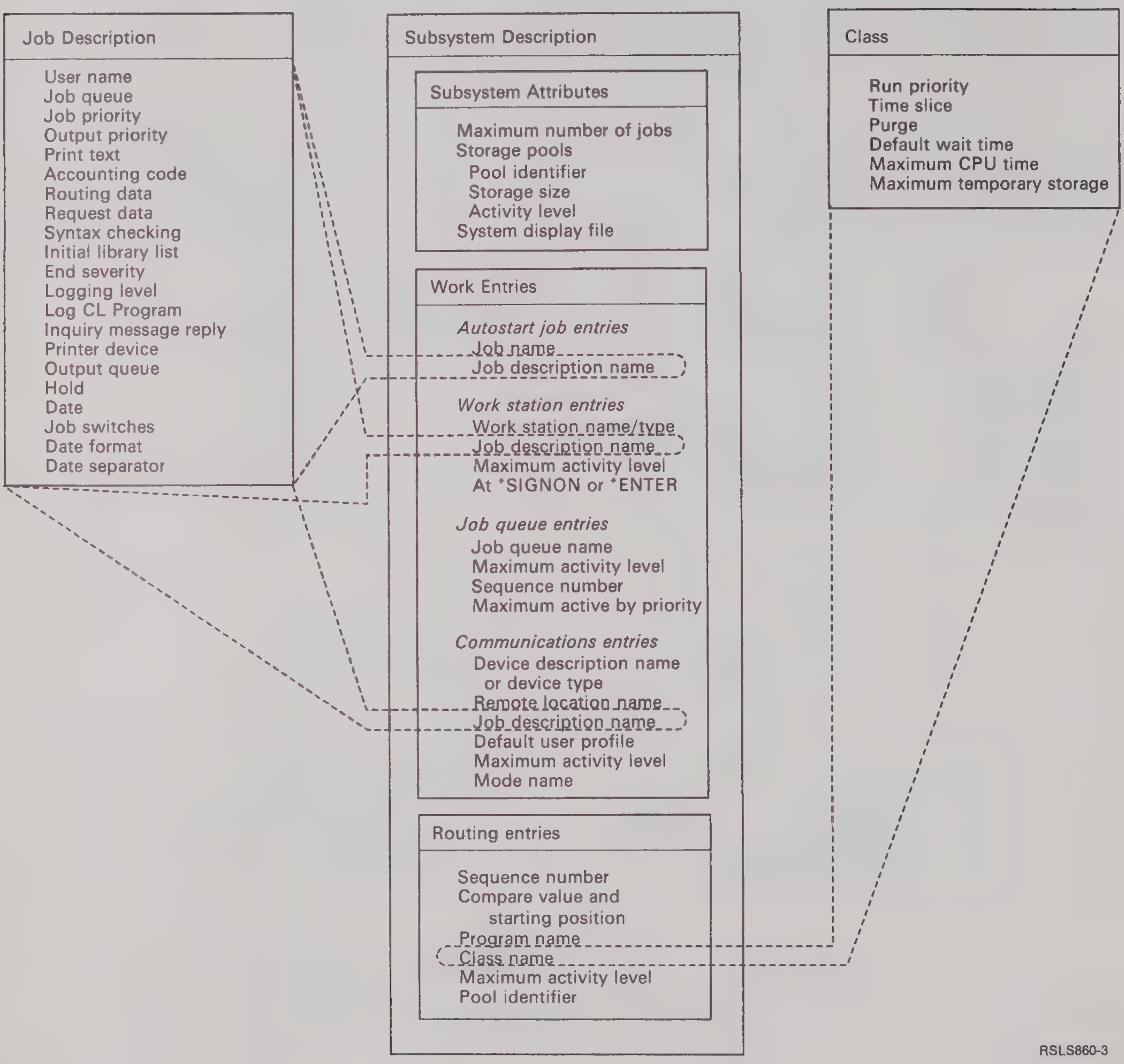
RSLS859-3

The following chart shows how the commands used to define a subsystem description operate on the created subsystem description.


CRTSBSD
The Create Subsystem Description command creates a new subsystem description.
ADDAJE
The Add Autostart Job Entry command defines a new autostart job entry.
ADDWSE
The Add Work Station Entry command defines a new work station entry.
Subsystem Description
Autostart Job Entries
Job Queue Entries
Work Station Entries
ADDCMNE
The Add Communications Entry command defines a new communications entry.
Communications Entries
Routing Table
ADDRTGE
The Add Routing Entry command defines a new entry in the routing table.
ADDJOBQE
The Add Job Queue Entry command defines a new job queue entry.

P7730187-0

The following chart shows how the job description and class relate to the subsystem description.


Job Description
User name
Job queue
Job priority
Output priority
Print text
Accounting code
Routing data
Request data
Syntax checking
Initial library list
End severity
Logging level
Log CL Program
Inquiry message reply
Printer device
Output queue
Hold
Date
Job switches
Date format
Date separator
Subsystem Description
Subsystem Attributes
Maximum number of jobs
Storage pools
Pool identifier
Storage size
Activity level
System display file
Work Entries
Autostart job entries
Job name
Job description name
Work station entries
Work station name/type
Job description name
Maximum activity level
At *SIGNON or *ENTER
Job queue entries
Job queue name
Maximum activity level
Sequence number
Maximum active by priority
Communications entries
Device description name
or device type
Remote location name
Job description name
Default user profile
Maximum activity level
Mode name
Routing entries
Sequence number
Compare value and
starting position
Program name
Class name
Maximum activity level
Pool identifier
Class
Run priority
Time slice
Purge
Default wait time
Maximum CPU time
Maximum temporary storage
RSLS860-3

Job Starting and Routing

A job can enter the system in one of seven ways:

- Interactively by signing on to a work station
- Interactively by transferring to a secondary job
- Interactively by transferring to a new group job
- Interactively by pressing the Test Request key
- In batch by submitting a job to a job queue
- In batch by issuing a communications program start request
- In batch by starting a subsystem with an autostart job in its subsystem description

When a job is started, the subsystem that is starting the job must determine what the first program is that the job will run. This is called *routing the job*. As was shown under "Routing Entries" on page 2-12, this is done by attempting to match up the routing data for the job with a routing entry in the subsystem description. Each time that a job causes a subsystem to search the routing table (for any reason) is called a *routing step*. Most jobs will only have one routing step, but it is possible for some jobs to have many routing steps. The following list identifies all of the possible ways to cause the job to have another routing step:

- Reroute Job (RRTJOB). This causes the job to reroute in the subsystem that the job was started in.
- Pressing F3 from the highest level of the command entry display (only valid for interactive jobs).
- Return (RETURN) command from the highest program in the job's program stack (only for interactive jobs–if a batch job issues the Return command, the batch job will end).
- TFRJOB and TFRBCHJOB commands. This actually causes the job to be placed on a job queue and then rerouted in the subsystem that has the job queue allocated.

To summarize, a job can have one or many routing steps, but all of the routing steps belong to that job. Normally, a job has only a single routing step.

Each routing step in a started job operates under the control of the program called at the start of the routing step. The routing data received by the system determines the routing entry in which the program name can be found.

Note: The IBM-supplied routing program QCMD runs the initial program and menu for interactive users. If you define your own routing programs, you should consider running a user's initial program and/or menu.

Routing data can be specified in one of the following ways:

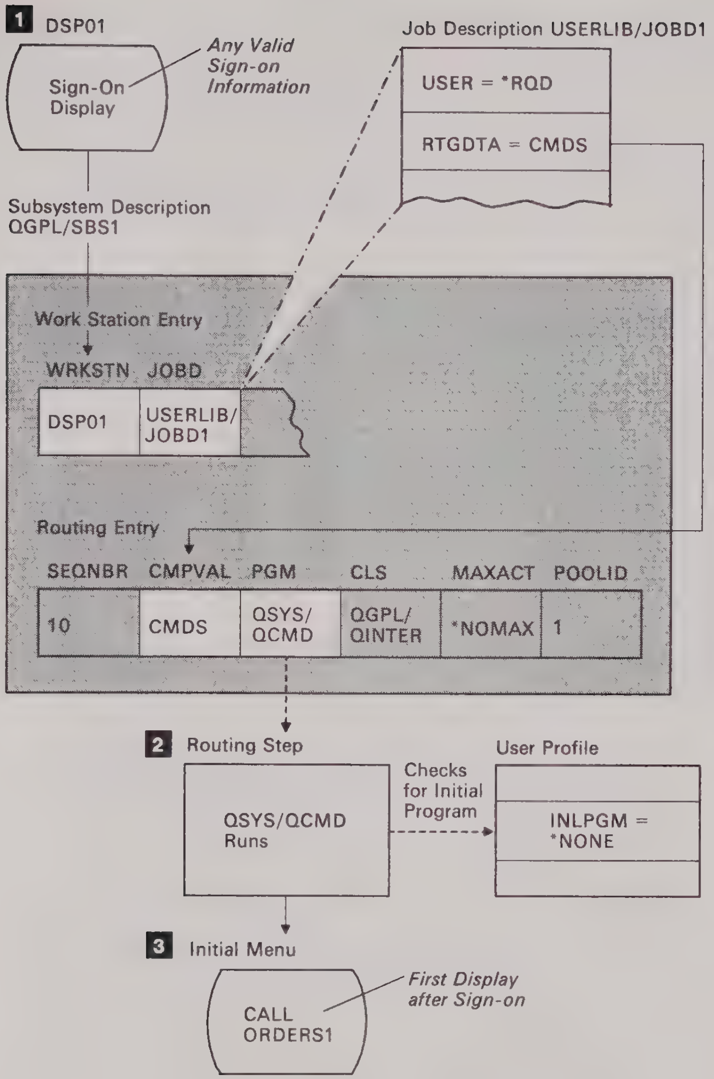
- In the job description for the job (RTGDTA parameter in CRTJOBD command), or RQSDTA parameter, if *RQSDTA is specified for the RTGDTA parameter.
- In the RTGDTA parameter specified in a BCHJOB or SBMJOB command (overrides the routing data in the job description) or RQSDTA parameter, if *RQSDTA is specified for the RTGDTA parameter.
- In the RTGDTA parameter specified on a Reroute Job (RRTJOB), Transfer Job (TFRJOB), or Transfer Batch Job (TFRBCHJOB) command (instead of the routing data in the job description) or RQSDTA parameter, if *RQSDTA is specified for the RTGDTA parameter. The RRTJOB, TFRJOB, and TFRBCHJOB commands can be run in a job after the job was initially routed using one of the previously described ways of specifying routing data.
- Routing data is created by the subsystem as a result of a communications program start request being processed. Routing data is based on information supplied by the remote system in the program start request (for example, program name and library).

For a list of all the commands that can be used with jobs, see the command grouping lists in the *CL Reference*.

Interactive Job Illustrations

Figure 2-2 on page 2-25 through Figure 2-6 on page 2-29 show techniques for building an environment in which a sample program USERLIB/ORDERS1 is called. (USERLIB/ORDERS1 could be an order entry program, an inquiry program, or a menu program.) The examples show the following programming techniques:

- Using the IBM-supplied program QSYS/QCMD to control the routing step
- Using a user program to control the routing step
- Routing based on user and routing based on work station
- Routing based on a communications program start request being processed by the subsystem


1 DSP01
Sign-On Display
Any Valid Sign-on Information
Subsystem Description QGPL/SBS1
Work Station Entry
WRKSTN JOBD
DSP01
USERLIB/ JOBD1
Job Description USERLIB/JOBD1
USER = *RQD
RTGDTA = CMDS
Routing Entry
SEQNBR CMPVAL PGM CLS MAXACT POOLID
10
CMDS
QSYS/ QCMD
QGPL/ QINTER
*NOMAX
1
2 Routing Step
QSYS/QCMD Runs
Checks for Initial Program
User Profile
INLPGM = *NONE
3 Initial Menu
CALL ORDERS1
First Display after Sign-on

1 A user signs on at work station DSP01.

The work station entry for DSP01 contains the name of the job description from which routing information is to be obtained.

The routing data CMDS matches the comparison value in the routing entry specifying program QSYS/QCMD.

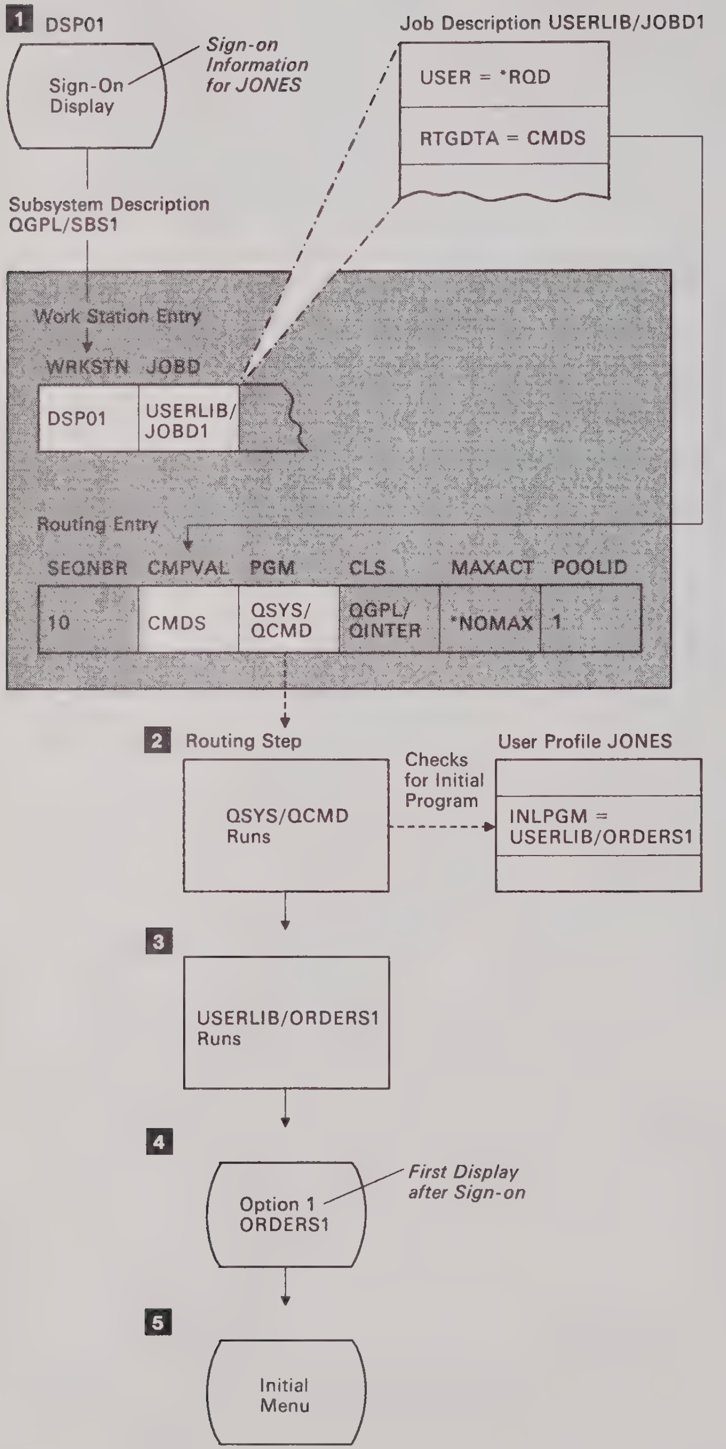
2 QSYS/QCMD is called for the routing step and checks the user profile for an initial program.

3 QSYS/QCMD displays the initial menu (no initial program specified).

The program USERLIB/ORDERS1 can be called from the main menu command line using the CALL command.

RSLS861-1

Figure 2-2. Interactive Jobs, Routing to QSYS/QCMD Based on Work Station


1 DSP01
Sign-On Display
Sign-on Information for JONES
Job Description USERLIB/JOBD1
USER = *RQD
RTGDTA = CMDS
Subsystem Description QGPL/SBS1
Work Station Entry
WRKSTN JOBD
DSP01 USERLIB/JOBD1
Routing Entry
SEQNBR CMPVAL PGM CLS MAXACT POOLID
10 CMDS QSYS/QCMD QGPL/QINTER *NOMAX 1
2 Routing Step
QSYS/QCMD Runs
Checks for Initial Program
User Profile JONES
INLPGM = USERLIB/ORDERS1
3
USERLIB/ORDERS1 Runs
4
Option 1 ORDERS1
First Display after Sign-on
5
Initial Menu
RSLS862-1

1 Jones signs on at work station DSP01.

The work station entry for DSP01 contains the name of the job description from which routing information is to be obtained.

The routing data CMDS matches the comparison value in the routing entry specifying program USERLIB/ORDERS1.

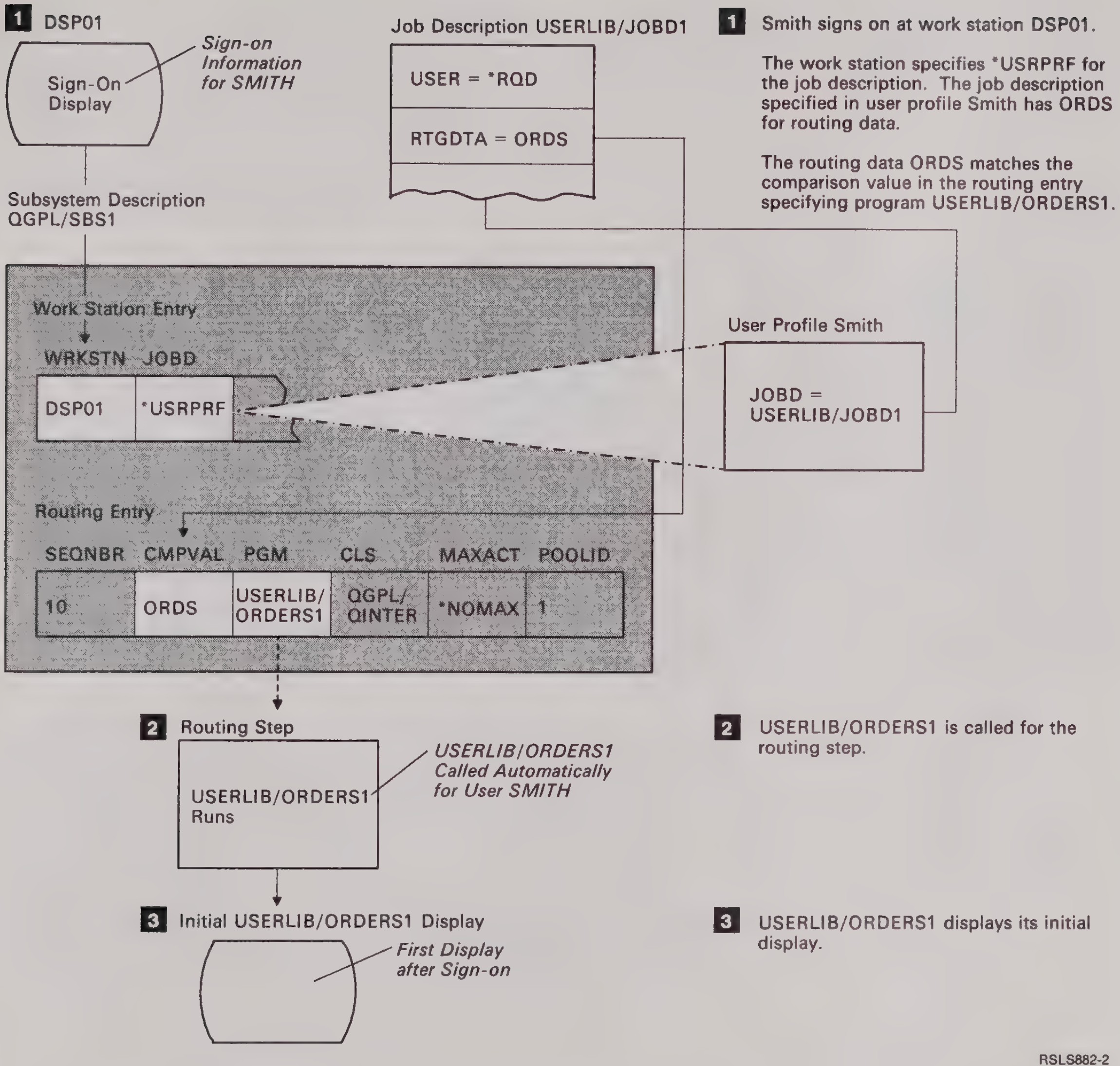
2 QSYS/QCMD is called for the routing step and checks the user profile for an initial program.

3 QSYS/QCMD calls the initial program (USERLIB/ORDERS1).

4 The display is presented by USERLIB/ORDERS1.

5 If ORDERS1 ends, the initial menu will be displayed. This can be prevented (if the user is only supposed to use ORDERS1) by specifying *SIGNOFF for INLMENU in the user's profile.

Figure 2-3. Interactive Jobs, Calling User Program Based on User Profile


1 DSP01
Sign-On Display
Sign-on Information for SMITH
Subsystem Description QGPL/SBS1
Work Station Entry
WRKSTN JOBD
DSP01 *USRPRF
Routing Entry
SEQNBR CMPVAL PGM CLS MAXACT POOLID
10 ORDS USERLIB/ ORDERS1 QGPL/ QINTER *NOMAX 1
Job Description USERLIB/JOBD1
USER = *RQD
RTGDTA = ORDS
User Profile Smith
JOBD = USERLIB/JOBD1
2 Routing Step
USERLIB/ORDERS1 Runs
USERLIB/ORDERS1 Called Automatically for User SMITH
3 Initial USERLIB/ORDERS1 Display
First Display after Sign-on
RSLS882-2

1 Smith signs on at work station DSP01.

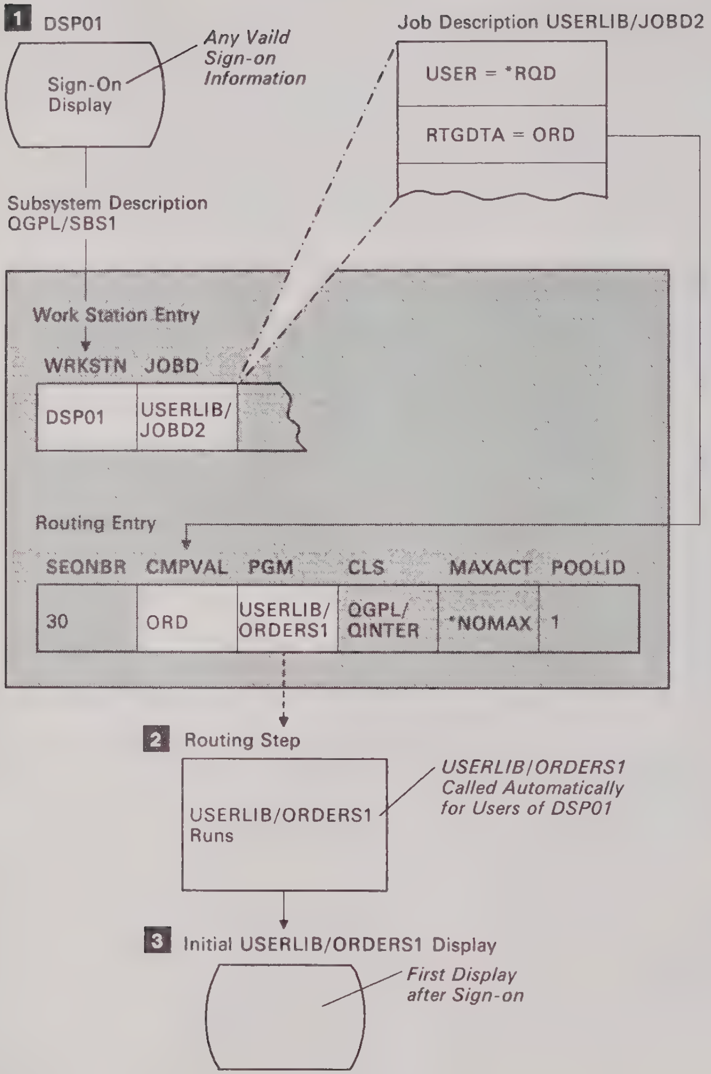
The work station specifies *USRPRF for the job description. The job description specified in user profile Smith has ORDS for routing data.

The routing data ORDS matches the comparison value in the routing entry specifying program USERLIB/ORDERS1.

2 USERLIB/ORDERS1 is called for the routing step.

3 USERLIB/ORDERS1 displays its initial display.

Figure 2-4. Interactive Jobs, Direct Routing to USERLIB/ORDERS1 Based on User


1 DSP01
Sign-On Display
Any Vaild Sign-on Information
Job Description USERLIB/JOBD2
USER = *RQD
RTGDTA = ORD
Subsystem Description QGPL/SBS1
Work Station Entry
WRKSTN JOBD
DSP01 USERLIB/JOBD2
Routing Entry
SEQNBR CMPVAL PGM CLS MAXACT POOLID
30 ORD USERLIB/ORDERS1 QGPL/QINTER *NOMAX 1
2 Routing Step
USERLIB/ORDERS1 Runs
USERLIB/ORDERS1 Called Automatically for Users of DSP01
3 Initial USERLIB/ORDERS1 Display
First Display after Sign-on
RSLS863-1

1 A user signs on at work station DSP01.

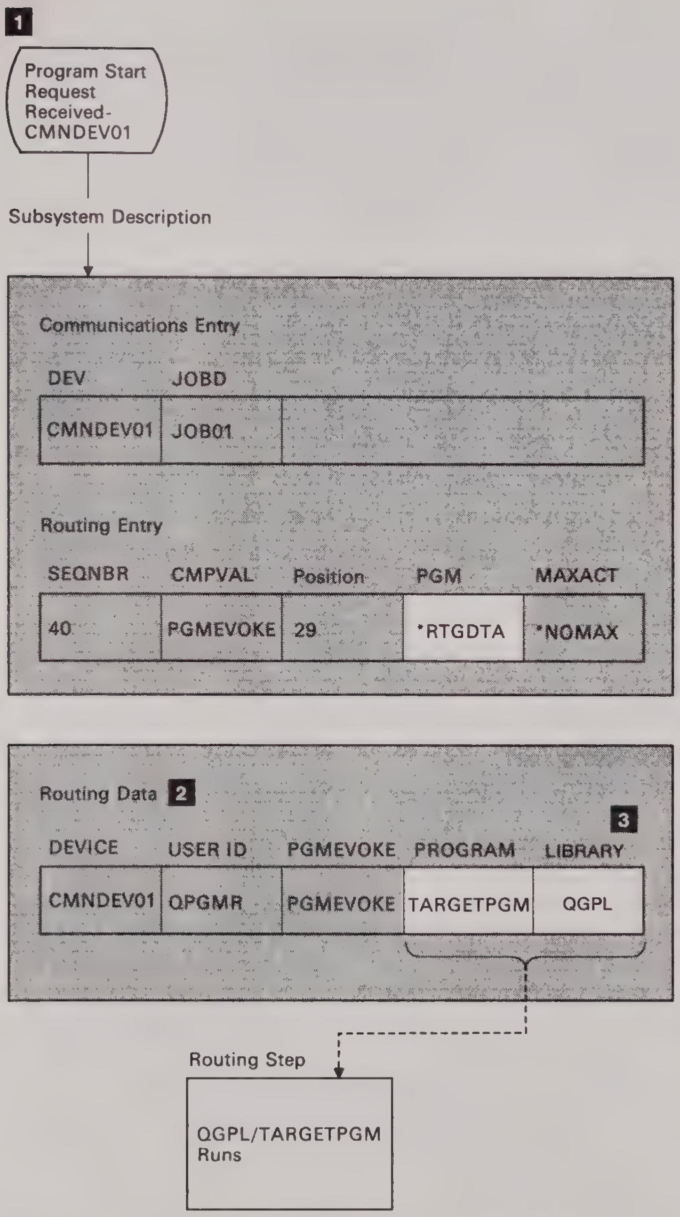
The work station entry for DSP01 contains the name of the job description from which routing information is to be obtained.

The routing data ORD matches the comparison value in the routing entry specifying program USERLIB/ORDERS1.

2 USERLIB/ORDERS1 is called for the routing step.

3 USERLIB/ORDERS1 displays its initial display.

Figure 2-5. Interactive Jobs, Direct Routing to USERLIB/ORDERS1 Based on Work Station


1
Program Start Request Received-CMNDEV01
Subsystem Description
Communications Entry
DEV
JOBD
CMNDEV01
JOB01
Routing Entry
SEQNBR
CMPVAL
Position
PGM
MAXACT
40
PGMEVOKE
29
*RTGDTA
*NOMAX
Routing Data
2
3
DEVICE
USER ID
PGMEVOKE
PROGRAM
LIBRARY
CMNDEV01
QPGMR
PGMEVOKE
TARGETPGM
QGPL
Routing Step
QGPL/TARGETPGM Runs
RSLS864-2

1 A program start request is received from the remote system.

2 The routing data for the program start request is created by the system based on the program start request information.

3 The program name specified on the program start request record is used, because the routing entry specifies to use the programs in the routing data.

Figure 2-6. Communications Batch Job Routing Based on Receiving a Communications Program Start Request

Note: If there is no library specified on the program start request, the subsystem library list is used to find the program.

End of Job Display

During an abnormal ending of an interactive job, the end of job display is shown. The end of job display contains the CPF message identifying the reason for the abnormal ending of the job. The display has a 30-second time-out; therefore, if the work station user does not press a valid key (Enter, Help, or a valid function key) in the time-out period, the display is removed when the time ends. Once the user returns from the help screen or the Display All Message screen, the timer is restarted.

The *CL Programmer's Guide* contains information on monitoring messages and creating a CL program to prevent a job from ending.

Interactive Job Considerations

The preceding interactive job illustrations show the basic ways in which interactive work can be handled. In determining the best approach for a particular job, you must first determine which program should control the routing step:

- The IBM-supplied command processor QSYS/QCMD gives the greatest flexibility in terms of making functions available to work station users. Using QCMD gives you the following benefits:
  - The attention program is activated if specified in the user profile.
  - The initial program specified in the user profile is called.
  - The initial menu specified in the user profile is called.
  - The user is placed in System/36 environment if specified in the user profile.

  By default, using QCMD brings you to the Main Menu where you can enter commands directly, including the CALL command to call user-written functions. Menu options with online help are provided to give easy access to system functions. Also provided are command selection menus, quick access to index search, and, if desired, the command entry function (called by CALL QCMD). The command entry functions are intended primarily for programmers and operators who require the full range of functions available through the direct use of commands.

- Your programs can be directly called to control the routing steps for interactive jobs. These programs can be designed to give a more specialized access to functions needed by your work station users than the IBM-supplied programs give. In addition, because your programs are tailored for specific functions, they should typically require even less system resource to support their running than the IBM-supplied programs. You may want to provide functions such as an initial program and initial menu also.

After you have determined which program controls the routing step, you must determine if routing is to be based on the work station from which the job was started, or on the user (user profile) who signed on. As shown earlier in this chapter, under "Interactive Job Illustrations" on page 2-24, routing based on the work station is accomplished using the routing data specified in the job description associated with the work station entry or profile for the device. Routing based on a user can be done using the initial program specified in the user profile or the job description in the user profile mapping to a routing entry other than QCMD.

Initial programs may interact with work stations to get input values from a work station user. When an initial program is called, it cannot receive parameter values.

An initial program can be used in one of two ways:

- It can be used to establish an initial environment for the user entering commands. For example, the library list can be changed or message files can be overridden so that messages in a non-English language can be used. When an initial program completes its function and returns to QSYS/QCMD, the initial menu is displayed.
- It can be used as the controlling program for the job. If the initial program does not return to QSYS/QCMD, it becomes the controlling program for the routing step. The initial menu is not displayed. The user can only request those functions available through the initial program.

  For example, a menu could be displayed with specific application options. The end user could only perform the functions on the menu. One of the options would normally be *sign off*. If the SIGNOFF command is run, the job ends and the system Main Menu is never displayed. If you use this approach, consider using the user profile option MENU(*SIGNOFF) to ensure no menu is displayed.

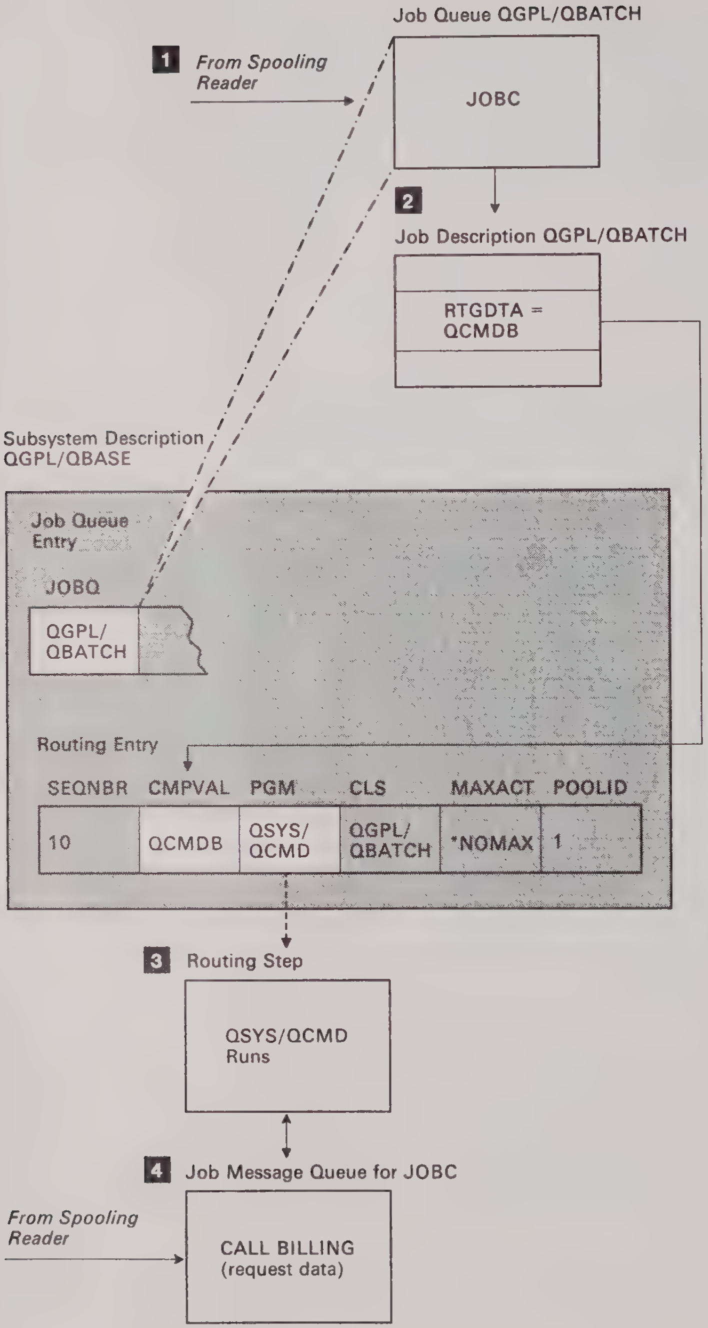
An initial program can be written so that when a return is issued, it either does or does not return to QSYS/QCMD. If the initial program returns to QSYS/QCMD, the initial menu is displayed. To avoid having QSYS/QCMD display the initial menu, the value *SIGNOFF can be specified for the initial menu in the user profile.

Batch Job Illustrations

Figure 2-7 on page 2-32 and Figure 2-8 on page 2-34 show two ways of routing batch jobs. The two ways are:

- Using the IBM-supplied program QSYS/QCMD to control the routing step of a job submitted through a spooling reader
- Using a user program to control the routing step for a job submitted using the SBMJOB command

In each illustration, the user program USERLIB/BILLING is called.


Job Queue QGPL/QBATCH
1 From Spooling Reader
JOBC
2
Job Description QGPL/QBATCH
RTGDTA = QCMDB
Subsystem Description QGPL/QBASE
Job Queue Entry
JOBQ
QGPL/ QBATCH
Routing Entry
SEQNBR CMPVAL PGM CLS MAXACT POOLID
10 QCMDB QSYS/ QCMD QGPL/ QBATCH *NOMAX 1
3 Routing Step
QSYS/QCMD Runs
4 Job Message Queue for JOBC
From Spooling Reader
CALL BILLING (request data)

1 The request data (commands) submitted for JOBC has been read by a spooling reader and placed on the job message queue for JOBC (CALL BILLING).

2 An entry for JOBC has been placed on the QGPL/QBATCH job queue by a spooling reader.

The JOB command for JOBC specified the job description QGPL/QBATCH.

The routing data in the job description matches the comparison value in the routing entry specifying program QSYS/QCMD.

3 QSYS/QCMD is called for the routing step and processes the request (commands) on the job message queue. QSYS/QCMD does not check the user profile for an initial program because an initial program is used only for interactive jobs.

4 The program USERLIB/BILLING is called when QSYS/QCMD processes the CALL BILLING command.

RSLS865-2

Figure 2-7. Batch Jobs, QSYS/QCMD Controls Routing Step for a Job Submitted through a Spooling Reader

An entry for JOBC was placed on the job queue QGPL/QBATCH by a reader. (The entry on the job queue contains a description of the job and scheduling information. The request data and any inline data are not included in the job queue entry.) The reader might have read the job from diskettes or a data base file. This particular job contains only request data consisting of commands. If the job had contained any inline data files, the reader would have spooled them so they could be used during job running. The input stream in the data base file was:

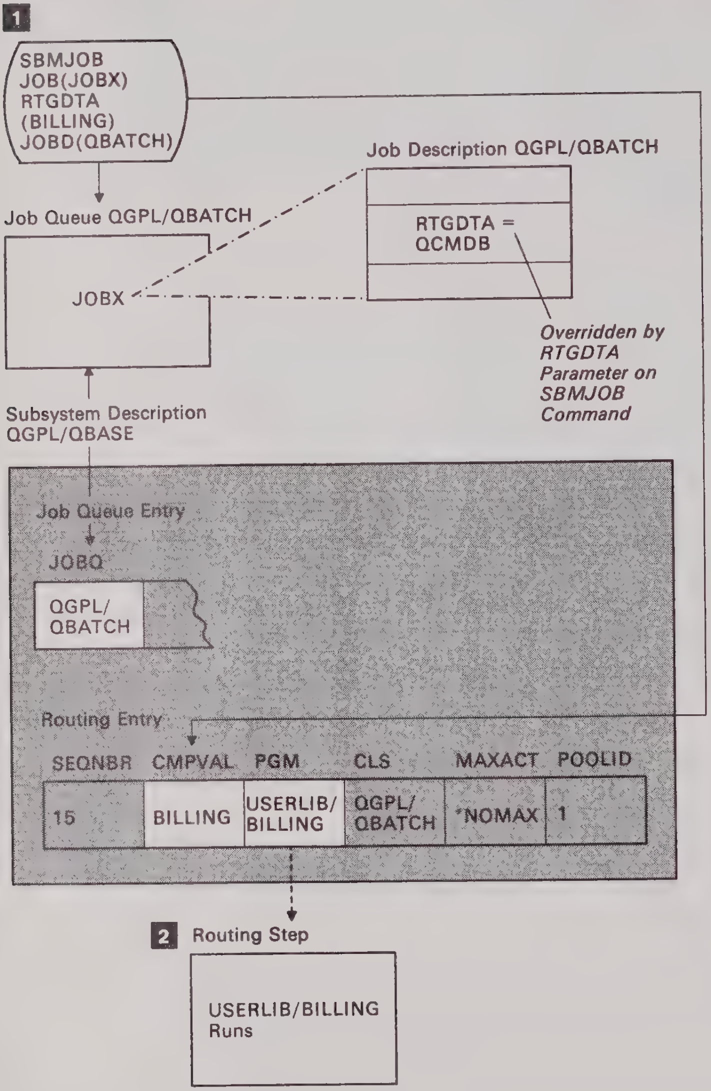
```
//BCHJOB JOB(JOBC) JOBD(QBATCH) JOBQ(QBATCH)
CALL BILLING
//ENDBCHJOB
```

The source for the request data is determined from the RQSDTA parameter on the BCHJOB command. The RQSDTA parameter can specify any one of the following:

- The actual request data (a character string for the parameter value)
- That the request data is in the job description (parameter value of *JOBD)
- That the request data follows the BCHJOB command (parameter value of *)

In Figure 2-8 on page 2-34, the RQSDTA parameter is not specified; it defaults to *, which means that the request data follows the BCHJOB command.

All other parameters on the BCHJOB command are not specified and default to the values in the job description QGPL/QBATCH.


1
SBMJOB
JOB(JOBX)
RTGDTA
(BILLING)
JOBD(QBATCH)
Job Description QGPL/QBATCH
Job Queue QGPL/QBATCH
RTGDTA =
QCMDB
JOBX
Overridden by
RTGDTA
Parameter on
SBMJOB
Command
Subsystem Description
QGPL/QBASE
Job Queue Entry
JOBQ
QGPL/
QBATCH
Routing Entry
SEQNBR
CMPVAL
PGM
CLS
MAXACT
POOLID
15
BILLING
USERLIB/
BILLING
QGPL/
QBATCH
*NOMAX
1
2
Routing Step
USERLIB/BILLING
Runs

1 An entry for JOBX has been placed on the QGPL/QBATCH job queue by using a SBMJOB command. (The SBMJOB command can be entered interactively as shown or can be in an input stream.)

The routing data in the job description QGPL/QBATCH is overridden by the RTGDTA parameter specified on the SBMJOB command **1**. (All other parameters on the SBMJOB command are not specified and default to the values in the job description QGPL/QBATCH or use values from the job that is running at the same time.)

The routing data specified (BILLING) matches the comparison value in the routing entry specifying program USERLIB/BILLING.

2 USERLIB/BILLING is called for the routing step.

RSLS866-3

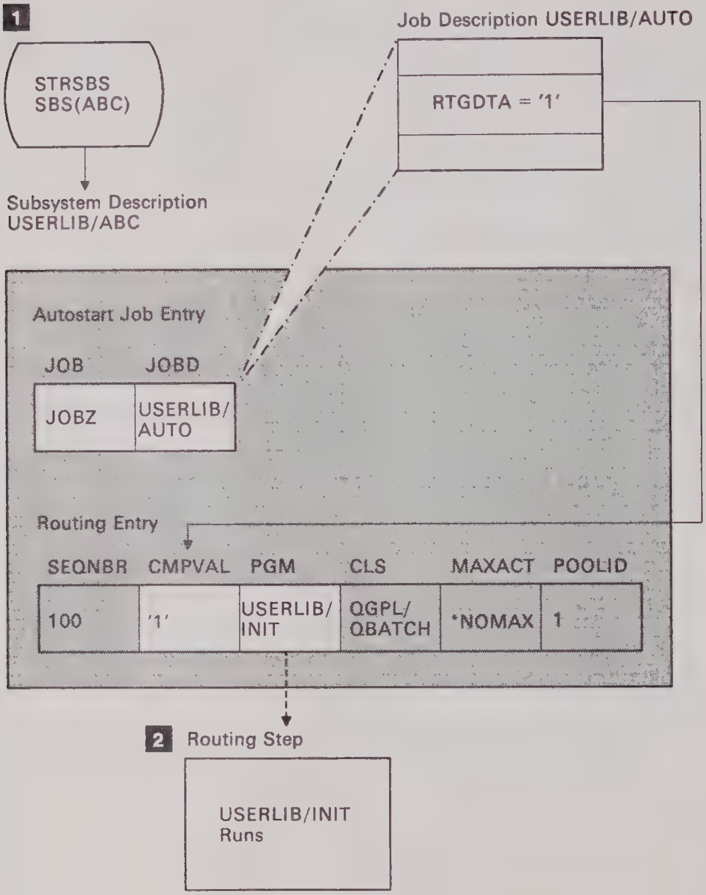
Figure 2-8. Batch Jobs, USERLIB/BILLING Controls Routing Step for a Job Submitted Using the SBMJOB Command

Batch Job Considerations

The preceding batch job illustrations show the basic ways in which batch work can be handled. The primary considerations in determining the best approach for a particular job are:

- Running QSYS/QCMD to process a stream of commands gives both the functional capabilities of CL and the flexibility of dynamically changing the CL source before submitting the job.
- Running QSYS/QCMD as the routing step program allows processing of a single command, for example, issuing the CALL command to call an application program. The called program may contain commands (including other CALL commands) to do the functions of an application. In most environments where changes are minimal, this approach offers the greatest flexibility with minimal system resource.
- Running your application program as the routing step program minimizes the amount of interpretive command processing. This is not normally significant, but may be worth considering if jobs are started very frequently and performance is critical.

Autostart jobs allow you to specify work that needs to be done each time a subsystem is started; for example, an autostart job can be used to start a spool writer. Figure 2-9 shows how autostart jobs are processed by the system.


1
STRSBS
SBS(ABC)
Subsystem Description
USERLIB/ABC
Job Description USERLIB/AUTO
RTGDTA = '1'
Autostart Job Entry
JOB
JOBD
JOBZ
USERLIB/
AUTO
Routing Entry
SEQNBR
CMPVAL
PGM
CLS
MAXACT
POOLID
100
'1'
USERLIB/
INIT
QGPL/
QBATCH
*NOMAX
1
2
Routing Step
USERLIB/INIT
Runs

1 STRSBS command has been issued specifying subsystem description USERLIB/ABC.

During the start subsystem processing, CPF checks the subsystem description for any autostart job entries.

The autostart job entry found specifies the job description USERLIB/AUTO.

The routing data from job description USERLIB/AUTO matches the comparison value in the routing entry specifying program USERLIB/INIT.

2 USERLIB/INIT is called for the routing step. The design of USERLIB/INIT determines whether it performs its function and ends or remains active until the subsystem ends.

Note: If more than one autostart job is specified for a subsystem, all autostart jobs are started immediately, not one followed by another. If the maximum activity level of the subsystem is exceeded, no other jobs can be started in the subsystem until enough autostart jobs have completed so that the number of jobs running is below the maximum activity level.

RSLS867-0

Figure 2-9. Autostart Job Processing

Note: The shipped QBASE and QCTL subsystem descriptions both use autostart jobs to start other subsystems and print writers.

Submitting a Batch Job from Another Job

You can submit a batch job in a running job (for example, submit a job from a display work station). The submitted job is placed on a job queue.

One way to submit a job in this manner is to use the Submit Job (SBMJOB) command. Another way is to use the Submit Data Base Jobs (SBMDBJOB) or Submit Diskette Jobs (SBMDKTJOB) command, that are described under "Job and Output Queues for Spooling" on page 2-17 earlier in this chapter.

Submitting a job with the SBMJOB command consists of specifying a job description (the job description specified in your user profile is the default) and, if necessary, overriding the parameters in the job description. Some of these parameters are from the submitter's job (for example, PRTTXT, INLLIBL, and so on). A job name can be specified, but if it is not, it is assumed to be the same as the simple job description name.

For example, the system operator can submit a job that saves a data base file. The file can be saved independent of other work the system operator has. The SBMJOB command looks like this:

```
SBMJOB     JOB(SAVDST) CMD(SAVOBJ ORDHDRP DSTPRODLB)
```

or

```
SBMJOB JOB(SAVDST) RQSDTA('SAVOB5 ORDHDRP DSTPRODLIB')
```

The JOBD parameter defaults to the job description specified in your user profile. The system-shipped defaults will cause the job to be placed on the job queue QGPL/QBATCH and routed to the IBM-supplied command processor QSYS/QCMD.

The USER parameter defaults to *CURRENT, which means that the job submitted is to be run under the same user profile that the submitter's job is running under.

The CMD (a command) is placed on the job message queue for the job submitted and is processed when QSYS/QCMD runs.

The RQSDTA parameter can also be used (mutually exclusive with CMD parameter) if a command string is to be submitted. The RQSDTA parameter should be used to construct a command or command keyword. For example:

```
SBMJOB RQSDTA('SAVOBJ' * BCAT '    ' OBJ(ORDHDRP)' *BCAT
'LIB(DSTPRODLIB)')
```

*USRPRF is the default for the MSGQ parameter. This means that a completion status message resulting from the submitted job will be sent to the user message queue specified in the user profile of the submitter.

Batch programs can also be submitted to a job queue from a CL program. The following example shows how to submit a batch program with a fixed parameter. The SBMJOB command is in a CL program. By placing a SBMJOB command in a CL program, you can free the work station user from the responsibility of remembering and entering all the parameters.

```
PGM
.
.
.
SBMJOB  JOB(JOBA)  JOBD(JOBA)  CMD(CALL  PROGX  PARM(DAILY))
SNDPGMMSG  MSG('Job  JOBA  submitted')  MSGTYPE(*COMP)
ENDPGM
```

The following example shows how to submit a program with character-type parameter values that are passed to a CL program.

```
PGM
.
.
.
DCL  &VAR1  TYPE(*CHAR)  LEN(6)
DCL  &VAR2  TYPE(*CHAR)  LEN(10)
.
.
.
SBMJOB CMD(CALL PROGX PARM(&VAR1 *BCAT &VAR2))
JOB(JOBA) JOBD(JOBA)
.
.
.
ENDPGM
```

If &VAR1 were XYZ and &VAR2 were ABC, the resulting command would be:

```
CALL PROGX PARM(XYZ ABC)
```

If &VAR1 were 123 and &VAR2 were ABC, the resulting command would be:

```
CALL  PROGX  PARM(123  ABC)
```

In this case, the first parameter is treated as a numeric parameter by the system, and PROGX must be coded as shown in the following text. If PROGX treats the parameter as a character value, then it is necessary to concatenate apostrophes into the RQSDTA parameter so that the resulting command would be:

```
CALL PROGX PARM('123 ƀ ƀ ƀ' ABC)
```

This can be done by making the following changes to the request data:

```
RQSDTA('CALL PROGX PARM(' *CAT '''' *CAT &VAR1 *CAT '''' +
        *BCAT &VAR2 *CAT ')')
```

The 4 apostrophes become a single apostrophe when the command is submitted. The outside 2 of the 4 apostrophes are the standard method of describing that what is inside is a literal. The inside 2 of the 4 apostrophes denote the system convention that 2 consecutive apostrophes will be recognized as a single-byte literal of an apostrophe.

If PROGX treats the first parameter as numeric (or if it was declared as numeric in the submitting program), then PROGX must declare the parameter as:

```
DCL &VAR1 TYPE(*DEC) LEN(15 5)
```

The length attributes (15 5) are the default attributes assigned by the system to a numeric parameter.

Submitting a Job Using Parameters from a Display File

As an example of how you can submit a batch job from another subsystem, the following describes a CL program that prompts for input from a work station user and passes the input received as parameters to a batch job. A display file is used to prompt the user.

The following is the display file (SBMJOBSMPD), which is used to prompt the user for the parameters to be passed:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00      A* SBMJOBSMPD - Submit data from display - used by SBMJOBSMPC
  2.00      A          R PROMPT
  3.00      A                                        CF03(93 'RETURN')
  4.00      A                                    1  2'Prompt for SMBJOB'
  5.00      A                                    3  2'Name'
  6.00      A            NAME         10   I      +2
  7.00      A                                    5  2'Department'
  8.00      A            DEPT          3   I      +2
  9.00      A                                    7  2'Days'
 10.00      A            DAYS          3  0I      +2
 11.00      A                                    9  2'Amount'
 12.00      A            AMOUNT        7  2I      +2
```

The following is the CL program (SBMJOBSMPC), which is used to display the SBMJOBSMPD file and submit the job:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
       /* SBMJOBSMPC - Submit job to batch with parameters                  */
                    PGM
                    DCLF       FILE(SBMJOBSMPD)
                    DCL        &DAYSA *CHAR LEN(3)
                    DCL        &AMOUNTA *CHAR LEN(8)
                    SNDRCVF
                    IF         (&IN93 *EQ '1') RETURN
                    CHGVAR     &DAYSA &DAYS
                    CHGVAR     &AMOUNTA &AMOUNT
                    SBMJOB     JOB(JOBA) JOBD(JOBA) RQSDTA('CALL SBMJOBSMP +
                                 PARM(' *CAT &NAME *BCAT '''' *CAT +
                                 &DEPT *BCAT '''' *BCAT &DAYSA *BCAT +
                                 &AMOUNTA *CAT ')') SWS('00000001')
                    ENDPGM
```

The program prompts for the parameters and receives data from the display file. The data is concatenated together to form the CMD parameter on the SBMJOB command.

The data from the display requires special processing because of the following rules:

- When you concatenate CL variables to make a parameter value, the CL variables must be character variables. You must also specify blanks, apostrophes, and parentheses that are to be included in the parameter to be passed.
- If you specify a parameter value starting with a digit (such as 43T) on a CALL statement, the system interprets the parameter as a numeric parameter.
- When a parameter is specified within apostrophes, and you want to specify another apostrophe within it, you must specify two apostrophes.

If the following information was specified on the prompt:

NAME	JONES
DEPT	57A
DAYS	5
AMOUNT	5000

the command submitted to the batch job queue would be:

```
CALL SBMJOBSMP PARM(JONES '57A' 5 5000)
```

Data is received from the display in the following fields:

- NAME: The work station user should key in all alphabetic data (no leading digits) for this field. It is received as a character variable in the CL program, and no special processing is required for the data.
- DEPT: The work station user could start with a digit when typing in the DEPT field (for example, 43T). If 43T is passed to the system on a parameter in a command, the system would treat the parameter as a numeric variable. This is the wrong data type; the DEPT parameter must be character data.

  To prevent this error, you must enclose the department number in apostrophes (such as '43T'). However, in the CL program, you must specify four apostrophes. When the CL program is compiled, the four apostrophes are processed by the system and become two apostrophes. When the CL program is run, the two remaining apostrophes are processed by the system again, and become one apostrophe.
- DAYS and AMOUNT: These numeric fields must be converted to character data so they can be concatenated into the RQSDTA parameter on the SBMJOB command. The program declares two character variables (DAYSA and AMOUNTA), then changes their value to the value of the numeric variables (DAYS and AMOUNT) as received from the display file. The program uses the character variables for the concatenation.

If the following information was specified on the prompt:

```
NAME      JONES
DEPT      43T
DAYS      2
AMOUNT    931
```

the request data sent to the batch job queue would be:

```
CALL SBMJOBSMP PARM(JONES '43T' 002 009.31)
```

The AMOUNT field contains a decimal point because it was changed from the numeric field AMOUNT to a character field by the CHGVAR command. The character variable must be long enough to contain this data.

The following is an example of how the SBMJOBSMP program would be coded to receive the parameters in batch if it were a CL program.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

                    PGM          PARM(&NAME &DEPT &DAYS &AMOUNT)
                    DCL          &NAME *CHAR LEN(10)
                    DCL          &DEPT *CHAR LEN(3)
                    DCL          &DAYS *DEC LEN(15 5)
                    DCL          &AMOUNT *DEC LEN(15 5)
                    DCL          &DAYSD *DEC LEN(3 0)
                    DCL          &AMOUNTD *DEC LEN(7 2)
                    CHGVAR       &DAYSD &DAYS
                    CHGVAR       &AMOUNTD &AMOUNT
                   .
                   .
```

P7730368-0

The program receives the character parameters with the desired definition. The numeric parameters must be received with a definition of LEN(15 5). (All numeric variables passed in from a processing program are passed as 15 5.) The data is then moved to the desired length fields by the CHGVAR command. If both parameters of the CHGVAR command are decimal variables, decimal alignment is performed.

The following is an example of how the SBMJOBSMP program would be coded to receive the parameters in batch if it were an RPG program:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

            C             *ENTRY     PLIST
            C                        PARM               NAME    10
            C                        PARM               DEPT     3
            C                        PARM               DAYS   155
            C                        PARM               AMOUNT 155
            C                        Z-ADDDAYS          DAYSD   30
            C                        Z-ADDAMOUNT        AMTD    72
            .
            .
```

P7730369-0

The program uses the same type of definition as the CL program to receive the parameters. The Z-ADD operation is used to convert the numeric data to the desired field definitions because it performs decimal alignment.

Rerouting and Transferring Jobs

Rerouting a job (RRTJOB command) starts a new job step in the same subsystem; transferring a job (TFRJOB or TFRBCHJOB commands) starts a new routing step in the same or another subsystem. In both cases, file overrides are removed, allocated objects except the work station and job message queue are deallocated, and files are closed. The library list stays the same because it is specified at the job level. The temporary library (QTEMP) and all objects in it remain the same, and all job attributes remain the same.

An interactive job is rerouted when either of the following occurs:

- The work station user presses the Exit key or enters the RETURN command from the only program in the program stack of a job. If QCMD is the only program on the program stack, an inquiry message warns the work station user about the rerouting because file overrides are lost, allocated objects are deallocated, and open files are closed.

  The work station user can continue by answering yes to the inquiry message, which will cause the job to be rerouted. By answering no to the inquiry message, the user can return to the command entry display.

- The work station user enters either the RRTJOB command or the TFRJOB command.

Rerouting a job can be used to change the processing attributes of a job. By rerouting a job, you can use a different routing entry to process the next routing step. For example, if the next part of a job needs to run in a different storage pool, you reroute the job so that it runs in a different storage pool.

If an error is detected during routing step starting, the routing step is started only to produce a job log and perform clean-up functions. The routing step started message is still sent and gives the user the name of the subsystem in which the routing step failure occurred.

Either an interactive job or a batch job can be transferred to another subsystem. However, to transfer a job, the user profile for the job must have *READ authority for the job queue and *USE authority for the subsystem description of the subsystem being transferred to.

A transferring interactive job enters the new subsystem through the job queue. To transfer an interactive job, the job queue being transferred to (representing the subsystem being transferred to) must belong to an active subsystem and there must be a work station entry for the work station in the subsystem description of the subsystem being transferred to. When an interactive job is placed on a job queue, it is given the highest job queue priority to minimize any delay. However, if the job queue is being held or if the maximum number of active jobs is already met for the subsystem, the transferring job must wait until the queue is released or until an active job ends. If the new subsystem is ended by the End Subsystem (ENDSBS), End System (ENDSYS), or Power Down System (PWRDWNSYS) command while a transferring interactive job is on a job queue, the job is ended as part of the subsystem ending.

A batch job can transfer itself to another job queue. In this case, the job queue need not be associated with an active subsystem, and it is placed on the job queue with its current job priority.

Note: If a batch job is on a job queue, it can be moved to a different job queue by specifying the JOBQ parameter on the CHGJOB command.

Spooled inline files can be accessed only during the first routing step of a batch job. If a batch job contains a Transfer Job (TFRJOB), Reroute Job (RRTJOB), or Transfer Batch Job (TFRBCHJOB) command the spooled inline files cannot be accessed in the new routing step.

Because a PWRDWNSYS command prevents new jobs or routing steps from being started by any subsystem, a batch job that has been placed on a job queue by a TFRJOB command may not complete before the system is powered down. The temporary objects associated with a transferring job (such as the library list and the QTEMP library and all objects in it) are cleared during a power down; therefore, during the initial program load (IPL), the system is unable to restore the job to its previous state. During the IPL, the system removes the job from the job queue and produces its job log.

However, if a batch job has been placed on a job queue by the Transfer Batch Job (TFRBCHJOB) command, the system saves the information that is necessary to start the batch job again if it is on the job queue during an IPL.

The temporary library (QTEMP) is cleared by the TFRBCHJOB command at each routing step. When the job is restarted after an IPL, it uses different internal temporary objects but they are initialized to data saved when the job was put on the queue by means of the TFRBCHJOB command. This includes having the same job name and a single job log made for the job.

The routing program is often used only once per job. If it calls another program and that program eventually does a return to the routing program, the following occurs:

- If QCMD is the routing program, the initial program is called (if any), and the initial menu is shown unless *SIGNOFF is specified in the user profile.
- If a call is from a user-written routing program, the return occurs as normal to the next instruction.
- If a user-written routing program does a TFRCTL, when the return occurs the routing program is called again.

The *CL Programmer's Guide* discusses how to write a program.

Summary of Controls on Levels of Activity

The maximum activity level is the number of jobs that can compete for machine time. The following specify levels of control over the starting of jobs:

- A subsystem. The MAXJOBS parameter on the CRTSBSD and CHGSBSD commands is used to specify how many jobs can be active at the same time in a subsystem. For an active subsystem, the sum of all jobs that are active at the same time started through work entries in the subsystem cannot exceed the MAXJOBS parameter value, except autostart jobs, which may temporarily cause the limit to be exceeded when the subsystem is started.
- A job queue entry. The MAXACT parameter on the ADDJOBQE and CHGJOBQE commands is used to specify how many jobs from a job queue can be active at the same time in the subsystem. (A MAXACT of 1 for a job queue forces jobs to be selected serially by job priority from a job queue.) The MAXPTYN parameter is used to specify how many jobs can be started for a specified job priority.
- A work station entry. If the WRKSTNTYPE parameter is specified, the MAXACT parameter on the ADDWSE and CHGWSE commands is used to specify how many jobs can be active at the same time in the subsystem for that entry.
- A communications entry. The MAXACT parameter on the ADDCMNE and CHGCMNE commands is used to specify how many jobs can be active at the same time for that entry.
- A routing entry. The MAXACT parameter on the ADDRTGE and CHGRTGE commands is used to specify how many routing steps can be active at the same time for that entry.

For the MAXJOBS and MAXACT parameters, active jobs refer to jobs that have started running but have not completed running. Active jobs do not include jobs that are on a job queue waiting to be started, or jobs that have completed processing but are waiting for a spooled file to be printed.

The following specify maximum levels to control the use of the processing unit for jobs which are already started:

- The system. The system value QMAXACTLVL (see Chapter 7, “System Values and Network Attributes”) is used to specify how many jobs can share main storage and processor resources at the same time. All active jobs (including system jobs) in all storage pools are controlled by QMAXACTLVL. The initial value of QMAXACTLVL is 100 jobs.
- The base storage pool. The system value QBASACTLVL (see Chapter 7, “System Values and Network Attributes”) is used to specify how many jobs can share the base storage pool at the same time and compete for use of the processing unit.
- User-defined storage pools. Use the POOLS parameter on the CRTSBSD and CHGSBSD commands to specify the activity level for user-defined main storage pools.

For the QMAXACTLVL, QBASACTLVL, and SBSD pool activity level, activity level refers to jobs sharing the processing unit at the same time. These jobs are all active jobs. When there are more jobs started than can run at the same time because of the activity level controls, the excess jobs have to wait to use the processing unit (normally this wait is very short).

For the storage pool activity level, the number of jobs running (or active jobs) refers to the number of jobs that have an activity level; that is, the jobs are actually running or they may be waiting for a disk I/O operation. In this sense, active jobs do not refer to jobs that are waiting for input, for a message, for a device to be allocated or for a file to be opened; or to jobs that are ineligible (jobs that are ready to run but the storage pool activity level is at its maximum). See Chapter 5, "Performance Guidelines," for a description of the states of a job in relation to the storage pool activity level.

The number of batch jobs that are allowed to start and the activity level specified for storage pools are the most useful. The storage pool activity level lets you limit the amount of running contention in the various storage pools in your subsystems. For the other activity limits, you would typically specify a value of *NOMAX unless special circumstances require that an actual limit be imposed. See Chapter 5, "Performance Guidelines," for a guide on how to determine the appropriate activity level for storage pools.

The following examples show the relationship of some of the activity controls. The system activity level is assumed to be 100.

Example 1. Two subsystems, SBSA and SBSB, use the base storage pool to run jobs. SBSA currently has two jobs running in this storage pool and SBSB has one. A work station entry in the subsystem description for SBSB specifies that any number of jobs can be started. The activity level of the base storage pool is 3. Therefore, only three jobs in the base storage pool can compete for the processing unit at a time. However, all the jobs are started.

Example 2. One autostart job, two work station jobs, and one batch job – four jobs in all – are in subsystem SBSC. The MAXACT for SBSC is specified as 4. No matter what is specified for the MAXACT of the work entries, no other jobs can be started until at least one job completes running.

Example 3. Subsystem SBSE is a batch subsystem for which 1 is specified for MAXACT. Although the job queue entry does not specify MAXACT, the limit is one job because 1 is the default for MAXACT for the subsystem. Therefore, jobs will be processed in job priority one at a time off the job queue.

Creating Your Own Subsystems

Creating Another Controlling Subsystem

You may want to create another subsystem description for the controlling subsystem with attributes different from those in the shipped controlling subsystem description QGPL/QBASE. IBM also ships a QGPL/QCTL controlling subsystem. If you want to use it instead of QBASE, you just change the QCTLSBSD system value.

You may change either QBASE or QCTL in QGPL. The system ships the QSYSSBSD subsystem in QSYS which you cannot change. However, the QSYSSBSD subsystem can be used in case your controlling subsystem becomes damaged or unusable.

If you create your own controlling subsystem description, you must change the system value QCTLSBSD. (QCTLSBSD contains the qualified name of the subsystem description for the controlling subsystem.) A change to this value takes effect at the next IPL unless the change is made by selecting “Define or change system at IPL menu” on the IPL options display during the IPL. In that case, the change takes effect during the current IPL.

The subsystem description for the controlling subsystem should contain a routing entry containing either *ANY or QCMDI as routing data, QSYS/QCMD as the program to be called, and class QSYS/QCTL or a user-defined class. This is because a user, usually the system operator, must be able to enter commands to do such things as free up storage if the auxiliary storage threshold has been reached.

The subsystem description for the controlling subsystem *must* contain a work station entry for the console device, and that entry must be of type *SIGNON. (*SIGNON is a value for the AT parameter, specified on the Add Work Station Entry (ADDWSE) command.) The *SIGNON value indicates that the Sign On display is displayed at the work station when the subsystem is started. This requirement ensures that the subsystem has an interactive device for entry of system and subsystem level commands. The End System (ENDSYS) command ends the OS/400 to a single session (or sign-on display) at the console in the controlling subsystem. A subsystem description that does not contain a work station entry for the console cannot be started as a controlling subsystem.

To have an alternative source of controlling input, the subsystem description for the controlling subsystem should have an entry for another work station. If a console problem is detected during an attended IPL and the system value QSCPFCONS is set to '1', the IPL will continue in unattended mode. Then, if the subsystem description for the controlling subsystem contains a work station entry for another work station, that work station can be used.

The subsystem description for the controlling subsystem should have a routing entry containing 525XTEST as routing data, QSYS/QARDRIVE as the program to be called, and QSYS/QCTL as the class. All work stations that are not allocated by subsystems or jobs have test request processed by the controlling subsystem.

You can use the subsystem description for the IBM-supplied controlling subsystem QBASE or QCTL as a model for creating your own controlling subsystem. You may want to leave the IBM subsystem unchanged as a backup. For more information on the subsystem description for QBASE and QCTL, see the *System Reference Summary*. You can use the CRTDUPOBJ command to create a duplicate of the QBASE or QCTL subsystem description.

After you have created the controlling subsystem, change the system value QCTLSBSD as follows (assuming the subsystem description is named QGPL/QCTLA):

```
CHGSYSVAL SYSVAL(QCTLSBSD) VALUE('QCTLA QGPL')
```

The change becomes effective at the next IPL.

If you create your own controlling subsystem, you should use a name other than QBASE or QCTL.

Interactive Subsystem Example

In the following example, objects are created for a subsystem in which interactive work is to be processed. The subsystem automatically allocates two work stations DSP01 and DSP02 when it is started. When a work station user signs on, program ORDLIB/ORDERPGM is called. Class QGPL/ORDCLS contains the running parameters for the routing step, and the job attributes are contained in the job description specified in the user profile.

The subsystem description ORDER is created and placed in library QGPL. When the subsystem ORDER is started, one storage pool with a size of 500K bytes is allocated. The pool's activity level is 2. The subsystem monitor programs called to handle the jobs in the subsystem ORDER use storage from pool 1, which is the shared base storage pool, but use machine pool activity level. The following CRTSBSD command creates the subsystem description ORDER:

```
CRTSBSD   SBSD(ORDER) POOLS((1 *BASE) (2 500 2)) +
          TEXT('Order entry jobs')
```

For an example of an autostart job in a controlling subsystem, see the example on "Calling a Special Recovery Program" on page 4-8, or the QBASE and QCTL subsystem descriptions in the *System Reference Summary*.

The following command creates the class ORDCLS and places it in QGPL:

```
CRTCLS   CLS(QGPL/ORDCLS) RUNPTY(45) TEXT('Class for order entry')
```

The following commands add two work station entries to the subsystem description QGPL/ORDER. When the subsystem ORDER is started, DSP01 and DSP02 are allocated to ORDER.

```
ADDWSE  SBSD(QGPL/ORDER) WRKSTN(DSP01)
ADDWSE  SBSD(QGPL/ORDER) WRKSTN(DSP02)
```

The following command adds a routing entry to the subsystem description QGPL/ORDER. The interactive jobs started from DSP01 and DSP02 are automatically routed to a routing step in which program ORDLIB/ORDERPGM is first called.

```
ADDRTGE SBSD(QGPL/ORDER) SEQNBR(010) CMPVAL(QCMDI) +
PGM(ORDLIB/ORDERPGM) CLS(QGPL/ORDCLS) POOLID(2)
```

The following command should be specified for any subsystem that has work station entries. Without it, you cannot use the Test Request key on the 5250 work stations allocated to the subsystem. (The Test Request key is used to start verification tests for a work station.)

```
ADDRTGE   SBSD(QGPL/ORDER) SEQNBR(900) CMPVAL(525XTEST) +
          PGM(QSYS/QARDRIVE) CLS(QGPL/QCTL)
```

Note: Because the POOLID parameter was not specified, a test request job runs in pool 1 of the subsystem ORDER. This pool is defined as the base storage pool.

Work Management Displays

The OS/400 provides commands you can use to display the contents and descriptions of work management objects. In addition, there are commands to display the status of activities in the system. You can use the following commands to monitor activity in the system:

- Display Class (DSPCLS)
- Display Job (DSPJOB)
- Display Job Description (DSPJOBD)
- Display Network Attributes (DSPNETA)
- Display Spooled File (DSPSPLF)
- Display Subsystem Description (DSPSBSD)
- Display System Value (DSPSYSVAL)
- Work with Active Jobs (WRKACTJOB)
- Work with Job (WRKJOB)
- Work with Job Descriptions (WRKJOBD)
- Work with Job Queues (WRKJOBQ)
- Work with Output Queues (WRKOUTQ)
- Work with Output Queue Descriptions (WRKOUTQD)
- Work with Readers (WRKRDR)
- Work with Spooled Files (WRKSPLF)
- Work with Submitted Jobs (WRKSBMJOB)
- Work with Subsystems (WRKSBS)
- Work with Subsystem Descriptions (WRKSBSD)
- Work with Subsystem Jobs (WRKSBSJOB)
- Work with System Status (WRKSYSSTS)
- Work with User Jobs (WRKUSRJOB)
- Work with Writers (WRKWTR)

For a description of how to use the system status display and the active jobs display, see Chapter 5, “Performance Guidelines.” For a detailed description of the other commands, displays, and their use, see the *CL Reference* and the *Operator's Guide*.

Security Considerations with Job Descriptions

Every job in the system operates under a job description. This controls the various attributes of a job. The USER parameter controls the name of the user profile that will be assigned to the job. There are several considerations to this parameter.

Interactive Jobs

The job description to be used is defined on the ADDWSE (Add Work Station Entry) command. The default is to use the job description specified in the user profile. If USER (*RQD) is specified in the job description, the user must enter a user name. If USER(XXXX) is entered (where XXXX is a specific user profile name), the user is allowed to press the Enter key on the Sign On display and operate under the XXXX user profile name.

Batch Jobs

The job description used for batch jobs is specified on the BCHJOB or Submit Job (SBMJOB) command.

If an input stream is entered containing BCHJOB commands, the user entering the STRXXXRDR or SBMXXXJOB command must have operational authority to the job description specified. When an input stream is used, jobs always operate under the user profile of the job description and not of the user who is placing the jobs on the job queue. If USER(*RQD) is specified in the job description, it is invalid to use the job description on a BCHJOB command.

If a SBMJOB command is used, the command defaults so that the batch job will operate under the user profile name of the submitter. However, if USER(*JOBD) is specified on the SBMJOB command, the job operates under the name in the job description. The submitter can specify USER(*JOBD) only if they have *CHANGE authority to the job description. (No authority is needed to the actual user profile.)

Frequently, a specific name in the job description is required in order to let users submit work for a specific user profile. For example, the QBATCH job description is shipped with USER(QPGMR) to allow this. This job description is created normally (the public is given *CHANGE authority). This means that any user on the system who has authority to the SBMJOB, STRXXXRDR, or SBMXXXJOB command can submit work under the QPGMR user profile. You may want to change the QBATCH job description depending on your security needs.

Security Concepts and Planning contains information on how to submit and use adopted authority.

Autostart Jobs

The job description used for an autostart job is specified on the ADDAJE (Add Autostart Job Entry) command. When the subsystem is started, the job operates under the user profile name in the job description, not under the name of the user who entered the ADDAJE command. Because the job description is located by qualified name at subsystem start time, you may want to control access to the job description. If USER(*RQD) is specified in the job description, it is invalid to use the job description for an autostart job.

Communications

The job description used for communications jobs is specified on the Add Communications Entry (ADDCMNE) command. It operates similarly to autostart jobs.

The ADDWSE, ADDAJE, ADDCMNE, and the corresponding CHGXXX commands require the user of the command to have *CHANGE authority to any job description named. It is valid to use these commands and name a job description that is not in existence at the time the command is entered. However, the job description name and library are set at that time.

Job Description Authority

To create a job description with a specific user name requires the user of the CRTJOBD or CHGJOBD (Create Job Description or Change Job Description) command to have *CHANGE authority to the user profile.

Periodically you may want to:

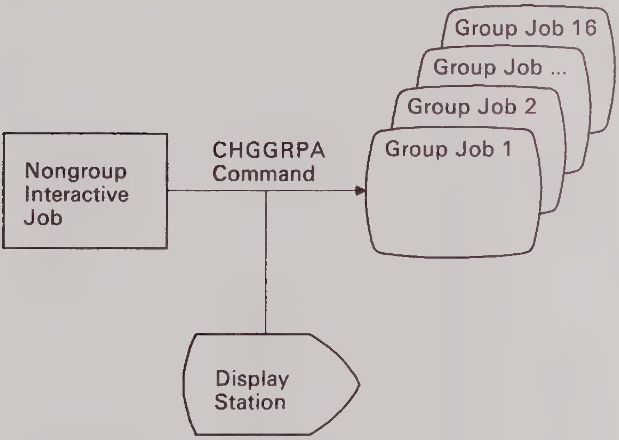
- Review all the ADDWSE commands and the job descriptions being used for interactive jobs to ensure that whenever job descriptions with specific names are used, they meet your security requirements.
- Review all your job descriptions that have a specific name in the USER parameter to determine if the authorizations to these job descriptions are meeting your security requirements. (If the job descriptions with specific names have public authority, then anyone can use them.) If you have security-sensitive job descriptions, you may want to place these in privately authorized libraries to help restrict their use.
- Review the authorization to user profiles to ensure that any object operational authorizations are consistent with your security requirements.

Chapter 3. Group Jobs and Attention Key Handling Programs

This chapter describes how to start many interactive jobs at one work station using group job support. How to start, handle, and end group jobs is described along with a sample application for group jobs. This chapter also describes Attention key handling programs which allow you to go quickly from one group job to another.

Concepts for Group Jobs

Group jobs support allows a user to start up to 32 interactive jobs at one work station. At any one time, only one group job can be active; the others are suspended. The group jobs are similar to secondary interactive jobs requested by pressing the System Request key; however, up to 16 group jobs can be started for each sign on at a work station (32 total when there is a secondary interactive job) and the application program can handle interruptions more flexibly.



P7730019-1

Attention key handling support makes it easy for the user to transfer control from one group job to another quickly, without ending one job to go to the other. To transfer control from one group job to another, the user presses the Attention key, and an Attention key handling program can either present a menu (from which the user chooses a group job) or immediately transfer the user to another group job.

The major advantages of group jobs are the following:

- The work station user can press the Attention key to interrupt work in one interactive group job, change to any of several other interactive group jobs, and return to the original group job quickly.

  The Attention key is made valid by the Set Attention Program (SETATNPGM) command and can be used independently of group jobs.

- Group jobs can give a significant performance advantage over alternative methods. If an Attention key handling menu is used to handle normal interruptions, the environment for processing these interruptions can be built on first use and then suspended. When a request is repeated, the function can be called with all files open and the proper screen displayed. This allows the process access group (PAG) for each function to be a minimum size, and the work station user does not have to exit a function to get to a menu from which to select another function.

- Using group jobs with display station pass-through provides a convenient and fast way to change among many interactive jobs on many different systems in a network. See the *Communications Programmer's Guide* for more information on display station pass-through.

The important concepts to understand about group jobs are:

- Group jobs apply only to interactive jobs.
- Up to 16 group jobs can exist in one group (16 more are available if the user transfers to a secondary interactive job).
- Group jobs are unique to a user (they are not shared by multiple users).
- Only one group job at a time is active (the others are suspended).
- Each group job is independent and has its own job log, spooled files, library QTEMP, and so forth.
- A group job is called by the Transfer to Group Job (TFRGRPJOB) command. This command is typically run from a user written menu program, which is called by pressing the Attention key (the SETATNPGM command must have been previously run).
- There is a 512-byte group data area that can be used to pass data between one group job and another. This group data area is implicitly created by the Change Group Attributes (CHGGRPA) command. The *CL Programmer's Guide* contains more information on group data areas.

Starting, Handling, and Ending Group Jobs

When working with group jobs, you can use the following commands:

- The CHGGRPA command changes your nongroup job to a group job and switches a group job back to a nongroup job (if it is the only job in the group).
- The TFRGRPJOB command switches from one group job to another group job in the same group and creates new group jobs. After each use of the TFRGRPJOB command, the SETATNPGM command must be used to set the Attention key on, if desired.
- The ENDGRPJOB command ends one group job in a group.
- The CHKRCDLCK command checks if the job has any record locks before transferring to another group job while you are in an update operation.
- The SIGNOFF command ends all group jobs in the group.

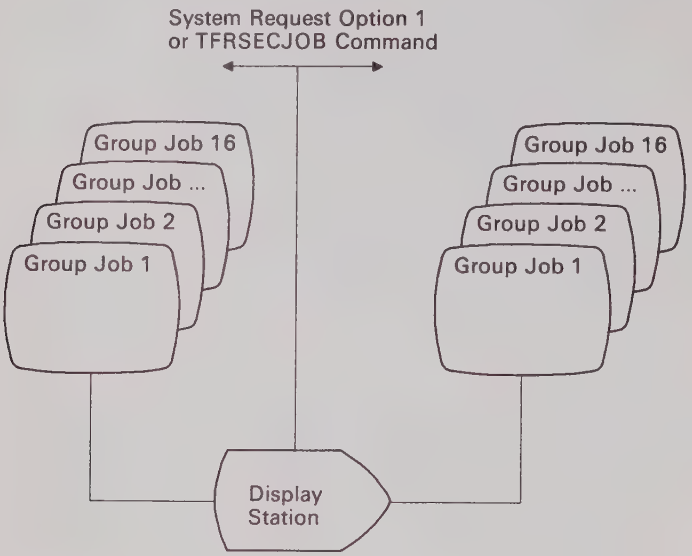
The CHGGRPA command identifies the current job as a group job and gives it a group job name to uniquely identify it in the group. (At this point the group has only one group job.) Each group job is unique for a user. Two different users do not share the same group job. When a job is designated as a group job, it then has the capability to call a new group job. There are also restrictions on group jobs (such as RRTJOB, TFRJOB may not be used). When there is only one active job in the group, that job can become a nongroup job.

To allow group jobs to communicate with each other, a special 512-byte data area called a group data area is automatically created when a job becomes a group job. The group data area can only be accessed by jobs in the group by using the special value *GDA.

The use of group jobs does not require an Attention key menu approach as described in this section. A group job can be called from any application program or by the GRPJOB(*SELECT) parameter on the TFRGRPJOB command.

Relationship of Group Jobs to a Secondary Interactive Job

The following shows the relationship of group jobs to a secondary interactive job.



P7730020-0

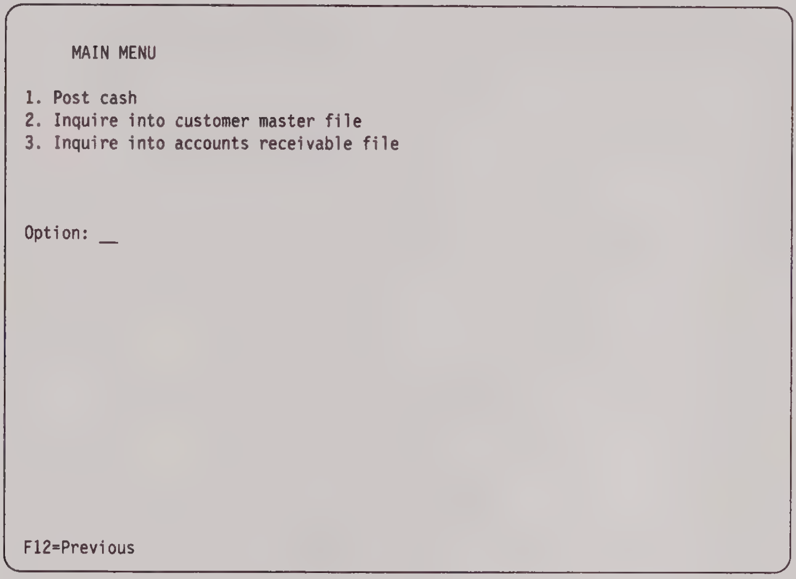
In this case, the CHGGRPA command is run twice, once for each interactive job.

The Group Job function is similar to the System Request function in that there is only one job active at a time while the others are suspended. Group jobs differ from system request in the following:

- Starting a group job does not require signing on. The same user profile and environment is used.
- Up to 16 group jobs can exist at any one time. The user must select which group job to transfer to, whereas using system request permits the user to transfer between only two jobs. Normally in group jobs, a menu reached by pressing the Attention key allows the user to select which group job to transfer to. It is possible to use group jobs together with system request for a total of 32 group jobs available for a single user. However, these 32 jobs are in two separate groups, each group having its own group data area and other group attributes.
- The System Request function allows the work station user to suspend a job while the keyboard is locked and application functions are in progress. This can interrupt a logical sequence of events. For example, records may be left locked. In contrast, the Attention key is active only when the keyboard is unlocked for input. Also, the application can control when the Attention key is active, and prevent its use at inappropriate times. The System Request function is always available if the work station user has authority to it.

Sample Application for Group Jobs

To understand the group job concept, assume a work station user is given a Main Menu with the following options:

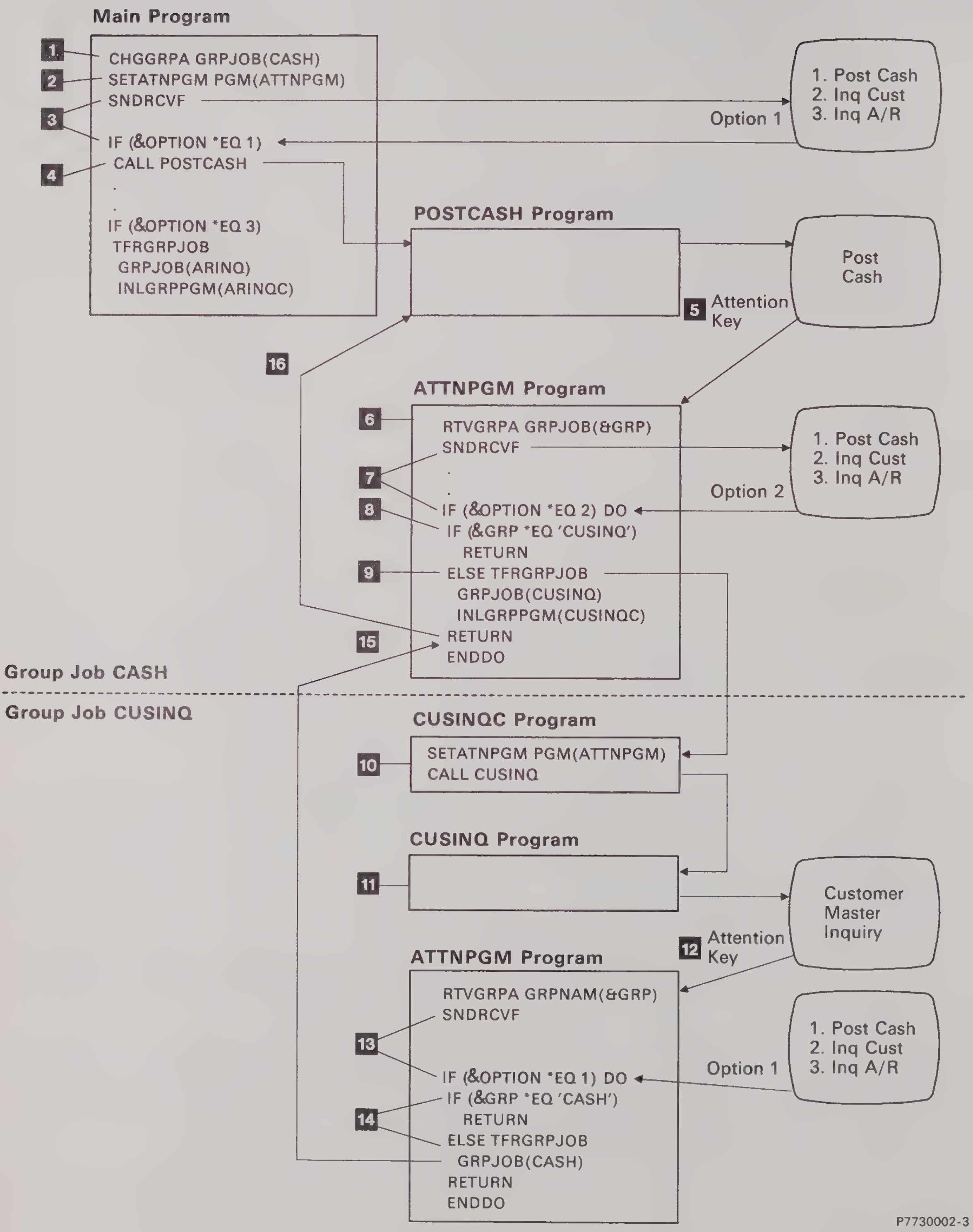


```
    MAIN MENU

1. Post cash
2. Inquire into customer master file
3. Inquire into accounts receivable file



Option: __










F12=Previous
```

The work station user selects option 1 (Post cash) to start entering payments into the file. While the work station user is entering data, he receives a telephone call requesting information that can be answered using option 2 (Inquire into customer master file).

There are several methods of handling such an interruption:

- Have the work station user end the Post cash function, return to the Main Menu, and select the Inquiry function. When done with the inquiry, the work station user returns to the Main Menu, selects the Post cash function and continues. This requires system interactions each time the work station user selects a different function and may require additional system resources for opening and closing files.
- Code the inquiry function as part of the Post cash function. This method is usually done by specifying that a command function key be enabled to request the inquiry. This can be awkward if the same Inquiry function must be accessed from multiple programs.
- Press the System Request key and start a secondary interactive job. This can be a good solution if the the work station user uses only one function during an interruption. However, if several functions are used, this solution can become awkward.
- Use the group job approach as shown in the following example:


Main Program
1
CHGGRPA GRPJOB(CASH)
2
SETATNPGM PGM(ATTNPGM)
SNDRCVF
3
IF (&OPTION *EQ 1)
4
CALL POSTCASH
.
.
IF (&OPTION *EQ 3)
TFRGRPJOB
GRPJOB(ARINQ)
INLGRPPGM(ARINQC)
1. Post Cash
2. Inq Cust
3. Inq A/R
Option 1
POSTCASH Program
Post Cash
5
Attention Key
16
ATTNPGM Program
6
RTVGRPA GRPJOB(&GRP)
SNDRCVF
.
.
7
IF (&OPTION *EQ 2) DO
8
IF (&GRP *EQ 'CUSINQ')
RETURN
9
ELSE TFRGRPJOB
GRPJOB(CUSINQ)
INLGRPPGM(CUSINQC)
RETURN
15
ENDDO
1. Post Cash
2. Inq Cust
3. Inq A/R
Option 2
Group Job CASH
Group Job CUSINQ
CUSINQC Program
10
SETATNPGM PGM(ATTNPGM)
CALL CUSINQ
CUSINQ Program
11
Customer Master Inquiry
12
Attention Key
ATTNPGM Program
RTVGRPA GRPNAM(&GRP)
SNDRCVF
13
IF (&OPTION *EQ 1) DO
IF (&GRP *EQ 'CASH')
14
RETURN
ELSE TFRGRPJOB
GRPJOB(CASH)
RETURN
ENDDO
1. Post Cash
2. Inq Cust
3. Inq A/R
Option 1
P7730002-3

The following steps illustrate how the sample application works:

1 The Change Group Attributes (CHGGRPA) command changes the current nongroup interactive job to a group job named CASH.

2 The Set Attention Program (SETATNPGM) command specifies that the system should recognize the Attention key and call the ATTNPGM program when the Attention key is pressed.

3 The SNDRCVF command displays a menu from which the work station user selects functions to work on. In this case, the work station user selects option 1 (Post cash).

4 The main program calls the POSTCASH program, and the work station user starts posting cash transactions.

Note: The main program runs a CALL command to the post cash program and a TFRGRPJOB command to the other functions. This allows the main job to always be known as CASH, but other techniques can be used. If the work station user chooses to *end* one of the group jobs other than CASH (instead of pressing the Attention key to suspend it), the system automatically returns to the previously active group job.

5 To change from posting cash, the work station user presses the Attention key. The system saves the current contents of the display and calls the Attention key handling program (ATTNPGM).

6 The Attention key handling program (ATTNPGM) is written to be used by any of the jobs in this group. The program retrieves the current group job name using the Retrieve Group Attributes (RTVGRPA) command.

7 The Attention key handling program displays a menu from which the work station user can select a function. The menu can contain any options, but in this example, the options are the same as those shown on the Main Menu. The work station user selects option 2 (Inquire into customer master file).

8 If the option is for the current group job (for option 2 this is CUSINQ), the program returns to the current group job. For example, assume the operator posting cash pressed the Attention key, then decided not to change group jobs and continue posting cash. Instead of transferring to a group job, only the RETURN command is needed.

9 If the option is for a different group job, the program runs a TFRGRPJOB command to suspend the current job and activate a new group job and program. For option 2, the TFRGRPJOB command specifies the name of the group job to be started (in this example, CUSINQ), and which program to call (CUSINQC program).

10 The group job CUSINQ is activated and the initial group program (CUSINQC) is called. The job has already been identified as a group job by the TFRGRPJOB command; therefore the CHGGRPA command is not needed. This group job also uses the SETATNPGM command to allow for an interruption. The same program (ATTNPGM) is used, but it runs in a different group job.

11 The work station user inquires into the customer master file and is now ready to return to the post cash function.

12 The work station user again presses the Attention key. The system saves the current contents of the display and calls the Attention key handling program (ATTNPGM). The user sees the same Attention Handling menu, but is in a different group job.

13 The work station user selects option 1 (Post cash).

14 Because the name of the current group job is CUSINQ (not CASH), the TFRGRPJOB command is run to transfer to the CASH group job.

15 In this instance, the group job is already active but suspended; therefore the system resumes the suspended group job. The ATTNPGM program is continued at the next instruction following the TFRGRPJOB command that was run in step 9. This causes a return to the POSTCASH program.

16 The previous display (which the system saved when the work station user pressed the Attention key in step 5) is restored, and the work station user continues posting cash transactions where he left off.

If the work station user wants to inquire into the customer master file again, he would again press the Attention key and receive the attention handling menu. Because the group job would already be active, the system would resume the job where it was suspended (that is, at the instruction following the TFRGRPJOB command in step 14). This return causes a return to the CUSINQ program where it was interrupted when the work station user pressed the Attention key in step 12.

If the work station user wants to inquire into the accounts receivable file, the same set of steps occurs. However, the work station user does not have to return to the post cash function first. The attention handling menu allows him to select any of the functions in any sequence. Note that the work station user cannot start two group jobs with the same group job name. If the job specified a group job name that is already active, control is passed to that job (no new group job is started) and the initial group program parameter is ignored. It is also possible to have several group jobs (having unique names) all doing the same function.

The Attention key cannot be pressed until the program requests input from the work station. This allows the program to prevent an interruption during certain functions; for example, when records are locked. See “Designing an Attention Key Handling Program” on page 3-13 later in this chapter.

When the Attention key is pressed, the system saves the current display. The current display is not saved when a TFRGRPJOB command is run.

The complete listing for this example is shown later in this chapter.

Effect of Call Level on Attention Key Status

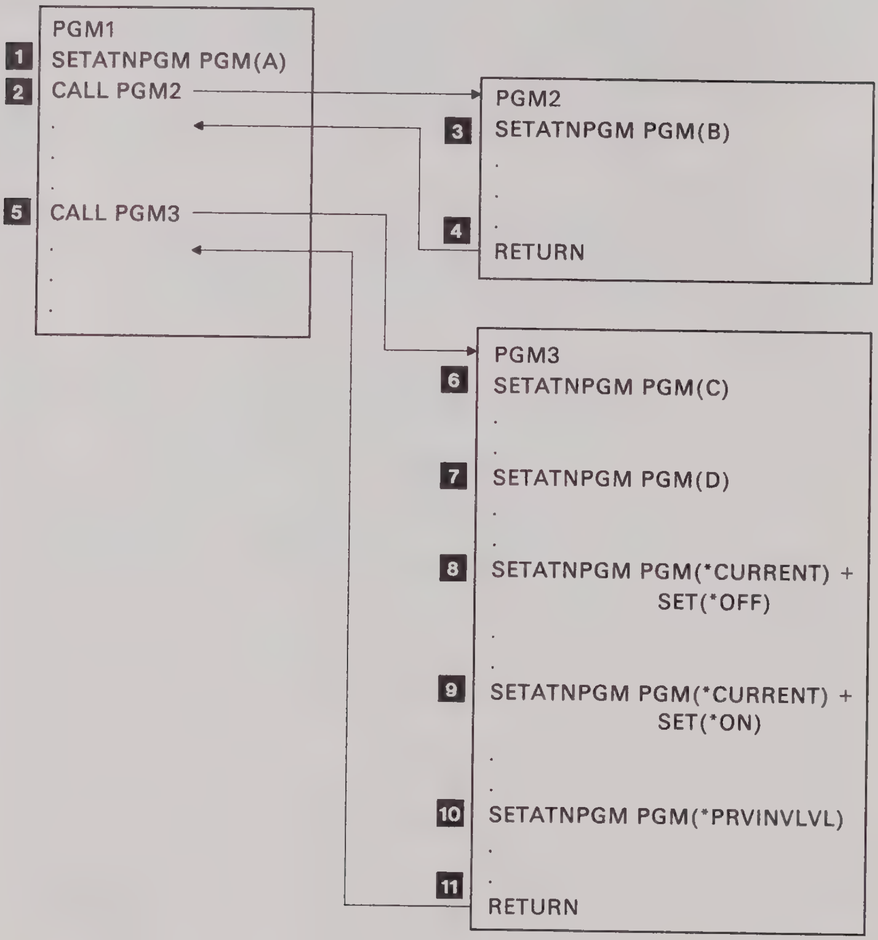
You can use the Set Attention Program (SETATNPGM) command with SET(*ON) specified to identify a program as the Attention key handling program at that call level in the job running the command. When the Attention key is pressed, the running job is interrupted, the display is saved, and the Attention key handling program is called. No parameters are passed to the Attention key handling program when it is called.

The Attention key handling program runs in the same job and has the same job attributes, overrides, and group authorities as the program that issued the SETATNPGM command. However, program-adopted authority is not propagated from the program that was interrupted.

The SETATNPGM command is call-oriented. That is, a SETATNPGM command issued at one call level causes the Attention key handling program to be in effect at the current call level as well as deeper call levels, until another SETATNPGM command is run to change the Attention key handling program or Attention key status. Whenever a program that issued a SETATNPGM command returns, the display is restored and the Attention key handling program and Attention key status are reset to what they were before the current call. If a Transfer Control (TRFCTL) command is used instead of a RETURN command, the status is not reset until the program that was transferred to returns.

You may also specify an Attention key handling program in the user profile. The *Security Concepts and Planning* contains information on this.

The following figure illustrates the use of the SETATNPGM command at different call levels:



```
   PGM1
1  SETATNPGM PGM(A)
2  CALL PGM2
   .
   .
   .
5  CALL PGM3
   .
   .
   .
```

```
   PGM2
3  SETATNPGM PGM(B)
   .
   .
4  .
   RETURN
```

```
   PGM3
6  SETATNPGM PGM(C)
   .
   .
7  SETATNPGM PGM(D)
   .
   .
8  SETATNPGM PGM(*CURRENT) +
                SET(*OFF)
   .
   .
9  SETATNPGM PGM(*CURRENT) +
                SET(*ON)
   .
   .
10 SETATNPGM PGM(*PRVINVLVL)
   .
11 .
   RETURN
```

P7730021-0

1 PGM1 issues a SETATNPGM command which causes program A to become the Attention key handling program.

2 PGM1 then calls PGM2. Program A continues to be the Attention key handling program until step 3.

3 Another SETATNPGM command causes program B to become the current Attention key handling program.

4 After returning from PGM2, program A again becomes the Attention key handling program.

5 PGM1 calls PGM3.

6 The first SETATNPGM command issued in PGM3 changes the Attention key handling program from program A to program C.

7 The second SETATNPGM command issued in PGM3 changes the Attention key handling program to program D.

8 Specifying SETATNPGM PGM(*CURRENT) SET(*OFF) causes no program to be called when the Attention key is pressed. This allows the program to complete a series of interactions with the work station user without being interrupted by the Attention key.

9 Specifying SETATNPGM PGM(*CURRENT) SET(*ON) causes program D to be called when the Attention key is pressed.

10 The Attention key handling program that was in effect at the previous recursion level goes into effect, that is, program A becomes the Attention key handling program again and the Attention key is set to *ON.

11 When PGM3 returns, program A continues to be the Attention key handling program.

Conditions for Using the Attention Key

In normal work station use, the Attention key can be pressed only when the keyboard is unlocked; that is, the program is ready for input. This occurs when a read or write-read operation is issued or the UNLOCK DDS keyword is used in a write operation. The use of the Attention key differs from that of the System Request key in that the application program has control over when it can be interrupted.

An exception to this occurs with application programs performing a get-no-wait operation on multiple device files. Pressing the Attention key causes these programs to be interrupted at any point by the Attention key handling program. (Even though the input inhibited light may be on, the keyboard is unlocked during a get-no-wait operation.) Application programs performing sensitive functions (especially during a get-no-wait operation) should therefore be protected by running SETATNPGM PGM(*CURRENT) SET(*OFF) before and SETATNPGM PGM(*CURRENT) SET(*ON) after sensitive code.

Note: A high-level language program can use the SETATNPGM command by calling QCMDEXC.

The Attention key cannot be used to call an Attention key handling program when the following conditions exist:

- The keyboard is locked. (Note the exception described earlier for get-no-wait operations.)
- The System Request menu or any of its options is being used.
- The display message display is shown.
- The OS/400 is already calling the Attention key handling program which makes it already active, however, if the program issues another SETATNPGM, the attention key will be enabled.
- A BASIC session is in progress, or a BASIC program is called.

Note: In a BASIC session, the Attention key is handled by BASIC, as appropriate. For example, if a BASIC program is called after a SETATNPGM command has set the Attention key on, the Attention key is handled by BASIC. After the BASIC program ends, your Attention key handling program takes effect again.

Guidelines for Coding Attention Handling Programs

Caution is necessary when defining an Attention key handling program because the Attention key handling program runs in the same job as the program that is in progress when the Attention key is pressed. Therefore, the interrupted program is not protected by any locks it held. If the interrupted program got an exclusive lock on an object, the Attention key program, because it runs in the same job, will be part of the job that has the exclusive lock.

The following guidelines are recommended for defining Attention key handling programs:

- Use simple functions such as menus that allow the work station user to transfer to another group job or to a secondary interactive job.
- Avoid referring to objects or functions that may be in use when the Attention key is pressed.
- Avoid calling nonrecursive functions when the Attention key is pressed. Nonrecursive functions are functions that cannot be interrupted, then called again. Many functions, such as high-level language programs and utilities like DFU, are nonrecursive.
- Avoid giving an option that allows the work station user to display the command entry display as part of the current job. For users who are programmers, it is meaningful to display a menu that includes an option for the command entry display. The command entry display should be specified as a separate group job (for example, by specifying INLGRPPGM(QCL) on the TFRGRPJOB command). This avoids re-using objects already in use.
- Attention key handling programs do not have the authority adopted by the program that was in progress before the Attention key was pressed.
- Be aware that a read-from-invited devices operation could time-out during the time that the Attention key handling program is running. Therefore, if a time-out were to complete in the program in progress while the Attention key handling program is running, whatever action taken as a result of that time-out will occur on return to the program in progress.

  For example, if the WAITRCD value in the file was set to 60 seconds and the program is set to exit if a key is not pressed in one minute, and the Attention key program is called and runs longer than that minute, the program will exit on return from the Attention key handler.

  However, caution should be used, since a check for available data is made before checking that the time-out has completed. If a key is pressed immediately after leaving the Attention key handler, data could be available that could complete the read-from-invited devices and the time-out would not be checked. This could cause unexpected results.

Designing an Attention Key Handling Program

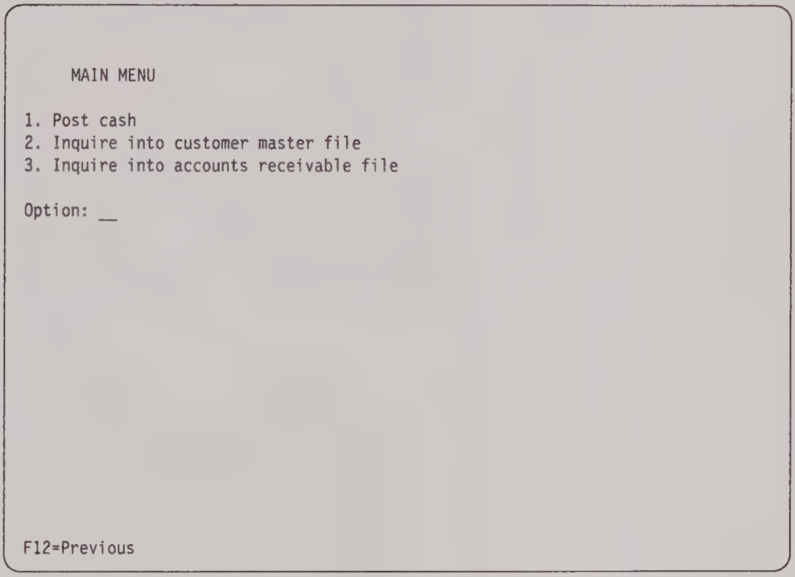
There are essentially three approaches to designing an Attention key handling menu:

- Fixed menu. The Attention key handling program allows only a fixed set of options.
- Dynamic menu. The work station user determines the options.
- Combination approach. The work station user selects from a fixed set of options and can dynamically select additional group jobs.

Fixed Menu

This is the method used in the example earlier in this chapter. In that example, the user can only select from the options on the menu.

A menu is displayed to the work station user with the following options:



```
    MAIN MENU

1. Post cash
2. Inquire into customer master file
3. Inquire into accounts receivable file

Option: __

F12=Previous
```

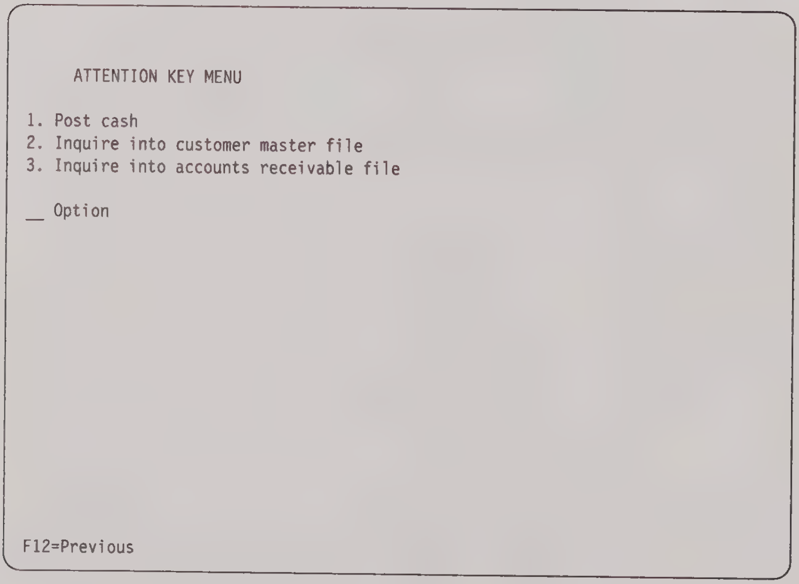
The program that displays the menu is coded as follows:

```
       PGM       /* First program */
       DCLF       MAIND
       CHGGRPA    GRPJOB(CASH) TEXT('Post cash')
       SETATNPGM  PGM(ATTNPGM)
LOOP:  SNDRCVF
       IF    (&IN91 *EQ '1') RETURN /* F12 */
       IF    (&OPTION *EQ 1) CALL POSTCASH
       IF    (&OPTION *EQ 2) TFRGRPJOB GRPJOB(CUSINQ) +
                             INLGRPPGM(CUSINQC) +
                             TEXT('Customer inquiry')
       IF    (&OPTION *EQ 3) TFRGRPJOB GRPJOB(ACRINQ) +
                             INLGRPPGM(ACRINQC) +
                             TEXT('Accounts Receivable inquiry')
       GOTO       LOOP
       ENDPGM
```

In this example, the CHGGRPA command changes the current job into a group job. The SETATNPGM command specifies that the Attention key is allowed, and it specifies the Attention key handling program to be called (ATTNPGM). Because this program will display a menu, it is called an Attention key menu program. If the work station user selects option 1 (Post cash), the program POSTCASH is called.

If another option is specified, a TFRGRPJOB command, which names the group job, is used. The system determines if the group job already exists. If it does not, the group job is started and the initial group program is used. If the group job already exists, the system transfers to that group job at the point where it was previously suspended. That is, it transfers to the next instruction following the TFRGRPJOB command that caused a transfer back to the CASH group job. The program loops back to the SNDRCVF command to redisplay the menu if control is ever transferred back to CASH.

When the work station user presses the Attention key, the Attention key menu program (ATTNNMU) is called. The display may differ from the initial menu, but in this example, they have the same options. Only the title differs to help the user identify the function he is using.



```
     ATTENTION KEY MENU

 1. Post cash
 2. Inquire into customer master file
 3. Inquire into accounts receivable file

 __ Option









 F12=Previous
```

The Attention key menu program (ATTNPGM) is coded as follows:

```
PGM       /* Attention key menu program */
DCLF      ATTNMNUD
DCL       &GRP *CHAR LEN(10)
CHKRCDLCK
 MONMSG MSGID(CPF321F)  EXEC(DO)
   SNDPGMMSG     MSG('Attention not allowed.  Complete transaction')
 RETURN
ENDDO
RTVGRPA   GRPJOB(&GRP)         /* Retrieve group name */
SNDRCVF
IF       (&IN91 *EQ '1') RETURN /* F12 */
IF       (&OPTION *EQ 1) DO     /* Post cash */
IF       (&GRP *EQ 'CASH') RETURN
         ELSE TFRGRPJOB GRPJOB(CASH)
         ENDDO
IF       (&OPTION *EQ 2) DO /* Cust inquiry */
         IF (&GRP *EQ 'CUSINQ') RETURN
         ELSE TFRGRPJOB  GRPJOB(CUSINQ) +
                         INLGRPPGM(CUSINQC) +
                         TEXT('Customer inquiry')
         ENDDO
IF       (&OPTION *EQ 3) DO /* Acct rec inquiry */
         IF (&GRP *EQ 'ACRINQ') RETURN
         ELSE TFRGRPJOB GRPJOB(ACRINQ) +
              INLGRPPGM(ACRINQC) +
              TEXT('Accounts Receivable inquiry')
         ENDDO
ENDPGM
```

The Attention key menu program may differ for any group job, but in this case the same program is used for all group jobs. Because the work station user may press the Attention key, then request the same function that was interrupted, the program retrieves the current group job name and compares it to the group job name for the option that was selected. If they are the same, a RETURN command is issued. This means that the work station user was in a function, pressed the Attention key, then decided to continue the function. If the names differ, the work station user is requesting a different group job and the TFRGRPJOB command is used. The system determines if the group job is already active; if it is not, the system starts the group job and calls the initial program. If the group job is already active, the system transfers to where it left off. Note that the CASH group job was originally made active and therefore the INLGRPPGM parameter is not needed.

When a group job is started, the system automatically establishes much of the environment by using the attributes from the transferring group job. Thus functions such as the library list and the logging level need not be set if the same values are desired. If however, the new group job also requires the use of the Attention key, it must be specified on a SETATNPGM command. For example, the initial program for the CUSINQ group job is CUSINQC as follows:

```
PGM         /* CUSINQC program */
SETATNPGM   PGM(ATTNPGM)
CALL        CUSINQ
ENDPGM
```

In the previous example, the same Attention key menu program is specified and then the processing program (CUSINQ) is called. If the work station user ends the CUSINQ program, the CUSINQC program also ends, and the system automatically ends the group job and transfers back to the group job that was previously active. It would be possible to issue a TFRGRPJOB GRPJOB(*PRV) after the CALL and then a GOTO to repeat the CALL command. This will keep the group job in a suspended state. This decreases the time necessary to transfer to this job on the next call of the CUSINQ option.

Dynamic Menu

In this case the Attention key menu program runs the TFRGRPJOB command with GRPJOB(*SELECT) specified. This provides a display of the current group jobs and allows the work station user to transfer to any group job. Thus the work station user can decide what functions are needed. This approach can be useful for programmers because it allows them to call several group jobs and to specify what command should be run.

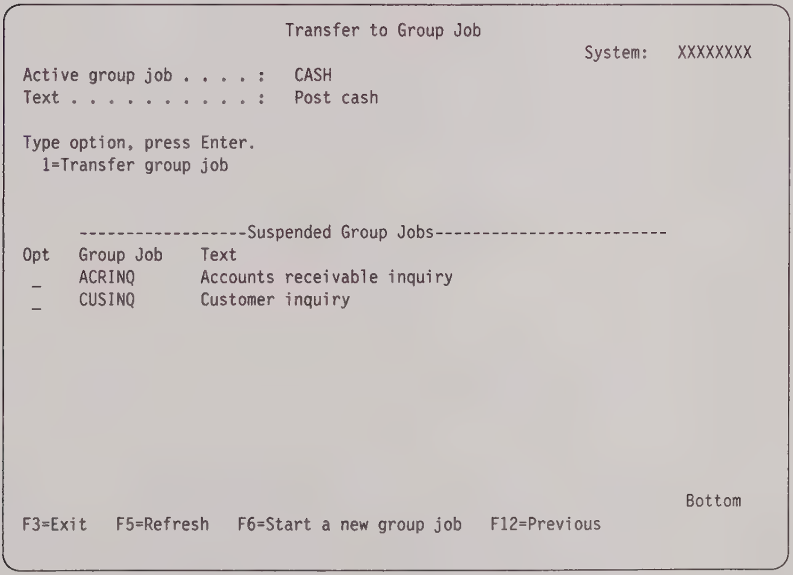
For example, assume that the work station user specifies the following commands or runs them as part of a standard setup program:

```
CHGGRPA     GRPJOB(NORMAL) TEXT('Normal job')
SETATNPGM   PGM(MENU1)
```

Program MENU1 would be coded as:

```
PGM
TFRGRPJOB   GRPJOB(*SELECT)
ENDPGM
```

Assume the work station user is working in the CASH group job, with two other group jobs suspended. If the work station user presses the Attention key, the display would appear as follows:



```
                              Transfer to Group Job
                                                                System:   XXXXXXXX
Active group job . . . . :   CASH
Text . . . . . . . . . . :   Post cash

Type option, press Enter.
  1=Transfer group job


      -----------------Suspended Group Jobs------------------------
Opt   Group Job    Text
 _    ACRINQ       Accounts receivable inquiry
 _    CUSINQ       Customer inquiry








                                                                       Bottom
F3=Exit   F5=Refresh   F6=Start a new group job   F12=Previous
```

The work station user could then press the F6 key to enter a TFRGRPJOB command with prompting. When the TFRGRPJOB prompt is displayed, the work station user could specify a GRPJOB parameter value and either of the following for the INLGRPPGM parameter:

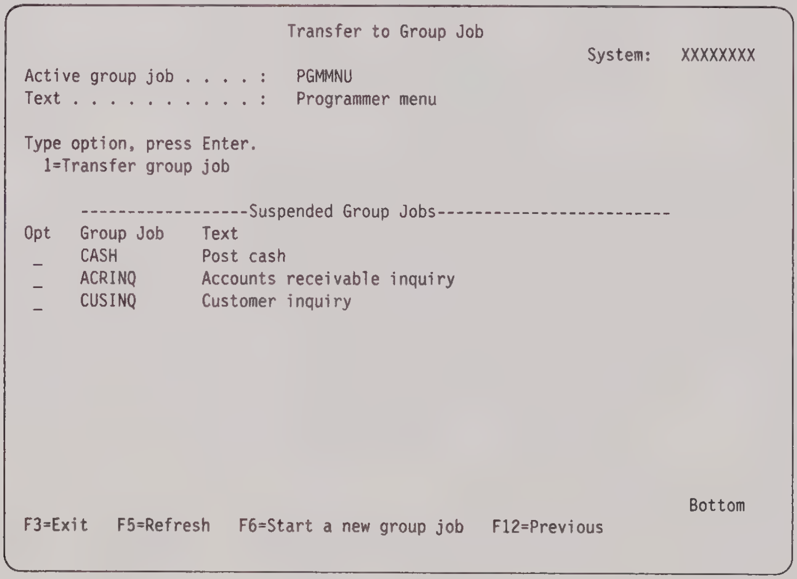
- QCMD, that would display the initial menu.
- A standard program that would build the environment the work station user wants and specify the same Attention key handling program. For example, assume the work station user specified the command:

```
TFRGRPJOB   GRPJOB(PGMMNU) INLGRPPGM(PGMMNUC) TEXT('Programmer Menu')
```

The PGMMNUC program could be coded as follows:

```
SETATNPGM   PGM(MENU1)
STRPGMMNU   .....
```

The work station user can now use the programmer menu. If the Attention key is pressed again, the display would show:



```
                              Transfer to Group Job
                                                            System:   XXXXXXXX
Active group job . . . . :   PGMMNU
Text . . . . . . . . . . :   Programmer menu

Type option, press Enter.
  1=Transfer group job

      -----------------Suspended Group Jobs------------------------
Opt   Group Job    Text
 _    CASH         Post cash
 _    ACRINQ       Accounts receivable inquiry
 _    CUSINQ       Customer inquiry







                                                                     Bottom
F3=Exit   F5=Refresh   F6=Start a new group job   F12=Previous
```

On this display, the jobs are shown in the order they were called.

Thus each group job that is already started can be easily called again, and the work station user can continue to add group jobs as required.

Combination Approach

This approach allows the following variations:

- The work station user can select from a fixed set of menu options. One option allows the work station user to prompt for the TFRGRPJOB command. The user could then specify the group job required or could specify GRPJOB(*SELECT) to receive a display of the currently active group jobs. If a user written menu is displayed, an input field on the menu for the group job name could also be considered. A modification of this is for the Attention key menu program to extract the active group jobs using RTVGRPA command and dynamically build the menu choices (including the standard choices).
- The work station user can select a menu option to activate a standard set of group jobs. Then, when the user presses the Attention key, the Attention key program runs the TFRGRPJOB command with GRPJOB(*SELECT) specified. This would display all the standard programs and allow the user to add group jobs.

The following section illustrates the second variation.

An Approach for Programmers

For programmers, you may want to set up several standard group jobs that are activated at sign on. The programmer can add additional group jobs if necessary. In the following example, the programmer has the following standard group jobs:

Description	**Group Job Name**
Main group job	MAIN
Second programmer menu	PGMMNU2
Command entry	CMDENT1
AS/400 Office word processing function	STRWP1

In this approach, each programmer has a unique initial program. The following shows an initial program for a programmer named SMITH:

```
                PGM
            1   CHGLIBL     LIBL(SMITH QGPL QPDA QTEMP QIDU)
                CHGGRPA     GRPJOB(MAIN) TEXT('Main group job')
               {CHGDTAARA   DTAARA(*GDA) VALUE('STRPGMMNU SRCLIB(SMITH) +
                               OBJLIB(SMITH) JOBD(SMITH)')
               {TFRGRPJOB   GRPJOB(PGMMNU2) INLGRPPGM(STDGRPC) +
                               TEXT('Programmer Menu # 2')
         2     {CHGDTAARA   DTAARA(*GDA) VALUE('CALL QCL')
               {TFRGRPJOB   GRPJOB(CMDENT1) INLGRPPGM(STDGRPC) +
                               TEXT('Command entry # 1')
               {CHGDTAARA   DTAARA(*GDA) VALUE('WRKPDMMBR FILE(SMITH/SMITH)')
               {TFRGRPJOB   GRPJOB(STRWP) INLGRPPGM(STDGRPC) +
                               TEXT('Display Write # 1')
            3   SETATNPGM   PGM(ATTNPGM)
LOOP:           STRPGMMNU   OBJLIB(SMITH) SRCLIB(SMITH) JOBD(SMITH)
                MONMSG      MSGID(CPF2320) /* F3 from Pgmrs Menu */
                GOTO        LOOP
                ENDPGM
```

RSLS869-3

This program does the following:

1 Establishes the environment (for example, library list).

2 Calls the standard group jobs (three group jobs plus the main group job are shown here). The group job data area is used to pass the command to be run. For each group job except the initial one, the same initial program (STDGRPC) is used.

3 The SETATNPGM command sets the Attention key on and establishes the ATTNPGM program as the standard Attention key handling program. When the Attention key is pressed, the TFRGRPJOB command with GRPJOB(*SELECT) specified displays the group job selection display. From this display, the programmer can change and add additional group jobs.

Note: To add another standard group job requires only two additional commands be added to the initial program and no changes to the other programs. With this approach the group jobs would be started at sign on and would not be ended unless the user signed off or entered the ENDGRPJOB command.

The initial program (STDGRPC) for the group jobs other than the main group job is as follows:

```
             PGM
             DCL         VAR(&CMD) TYPE(*CHAR) LEN(512)
           1 RTVDTAARA   DTAARA(*GDA) RTNVAR(&CMD)
             TFRGRPJOB   GRPJOB(*PRV)
           2 SETATNPGM   PGM(ATTNPGM)
 LOOP:     3 CALL        QCMDEXC (&CMD 512)
             MONMSG      MSGID(CPF2320) /* F3 from Pgmrs Menu */
             TFRGRPJOB   GRPJOB(*PRV)
             GOTO        LOOP
             ENDPGM
```

RSLS870-0

When the group job is started, it does the following:

1 Retrieves the group data area, which contains the command to be run. The command is not run immediately, but is stored in a variable in the program. The program returns to the previous group job. The user does not see any displays, but would notice the overhead to build the group job. Assume at a later point the user selects the group job SRTWP1 from the Group Jobs Selection menu. The user will transfer to the STRWP1 group job. Because it is already active, it continues with the instruction following the TFRGRPJOB command.

2 The SETATNPGM command sets the Attention key on and establishes the ATTNPGM program as the Attention key handling program.

3 The program calls the QCMDEXC program and passes variable &CMD, which contains the SRTWP command (see the initial program shown earlier). To exit the AS/400 Office word processing function, the programmer can do either of the following:

- Press the Attention key and receive the Group Jobs Selection menu. This allows the programmer to transfer to an existing group job or start a new group job.
- Press the Exit key from the AS/400 Office word processing function primary menu. This returns to program STDGRPC, which transfers back to the previous group job.

  Note: Pressing the Exit key does not end the SRTWP1 group job. If the STRWP1 group job is again activated, the program returns to the next instruction, which loops back and runs the STRWP command again by calling QCMDEXC.

The Attention key handling program (ATTNPGM) would be coded as follows:

```
PGM
TFRGRPJOB  GRPJOB(*SELECT)
MONMSG     MSGID(CPF1310) EXEC(DO) /* No group jobs */
SNDUSRMSG  MSG('An attention program is active, but no +
               group jobs exist') MSGTYPE(*INFO)
ENDDO      /* No group jobs */
ENDPGM
```

If the user presses the Attention key, and then presses the F6 key to start a new group job (one that is not in his standard list), he must enter the SETATNPGM command in the new group job in order for the Attention key to be active.

If the user wants to end all group jobs he can do so by calling the following program:

```
         PGM
         DCL        &GRPJOBL *CHAR LEN(1056)
         DCL        &X *DEC LEN(5 0) VALUE(67) /* 2nd group job */
         RTVGRPA    GRPJOBL(&GRPJOBL)
         MONMSG     MSGID(CPF1311) EXEC(DO) /* Not a group job */
         SNDPGMMSG  MSG('The current job is not a group job')
         RETURN
         ENDDO      /* Not a group job */
LOOP:    IF         (&X *NE 1057) DO /* Less than 16 group jobs */
         IF         (%SST(&GRPJOBL &X 10) *NE '      ') DO /* Job */

         ENDGRPJOB  GRPJOB(%SST(&GRPJOBL &X 10))
         ENDDO      /* Job */
         CHGVAR     &X (&X + 66)
         GOTO       LOOP
         ENDDO      /* Less than 16 group jobs */
         CHGGRPA    GRPJOB(*NONE)
         SNDPGMMSG  MSG('All group jobs ended and the +
                        current job is no longer a group job')

         ENDPGM
```

The program retrieves the current group job list, which can be made up of 16 entries where each entry is 66 bytes long containing:

Group job name	10 characters
Job number	6 characters
Group job text	50 characters

The active group job is the first one in the list. The program starts by trying to access the name of the second group job (starts in position 67) and if it is not blank, it ended the group job. When all other group jobs are ended, the current job is changed into a nongroup job.

Returning to the Group Job Main Program

In all previous examples, the work station user was allowed to go to any function from any other function. An alternative is to have the work station user always return to the main program. This could be done using the previous example program and removing the SETATNPGM command, as follows:

```
        PGM
        OVRDBF . . . SHARE(*YES)
        OPNDBF . . .
LOOP:   CALL  CUSINQ
        TFRGRPJOB  GRPJOB(*PRV)
        GOTO  LOOP
        ENDPGM
```

The work station user cannot use the Attention key. Ending the function (exiting from the CUSINQ program) returns the work station user to the main program.

As in the previous alternative, the only way to end this group job is to use the ENDGRPJOB command, use the ENDJOB command, or to sign off.

Note: Using the OVRDBF and OPNDBF commands to specify a shared open, allows a faster open each time a group job is called.

Transferring to Another Group Job without Seeing a Menu

You can use the Attention key to transfer directly to another job without seeing a menu. For example, the Attention key handling program for group job A could transfer to group job B. The Attention key handling program for group job B could transfer back to group job A. This allows a single keystroke to be used to switch between functions.

Ending Group Jobs

In some environments it may be desirable to force the end user to correctly end certain group jobs rather than issuing the ENDGRPJOB command. For example, assume that the user may have a group job where there is a complex update involved and you want to be sure the job is ended normally. Another example is where the user may be in the middle of a SEU session and should complete the function normally.

It is possible to achieve this with the support given by the system. For example, you could do the following:

- Set a switch in the group data area that could be tested by each of the group jobs to function as the shutdown switch. That is, when the switch is set on, the group jobs function should be ended.
- Access the active group job names by using the RTVGRPA command and the GRPJOBL return variable. Compare each name accessed (start with the second group job) against a predetermined list of the group job names that should be correctly ended. If the group job name is not in the list, it can be ended immediately by the ENDGRPJOB command. If the job must be correctly ended, transfer to the group job using the TFRGRPJOB command.
- The Attention key handling program for all group jobs would have to be sensitive to the shutdown switch and would prevent transferring to another group job if the switch is set on.

- If you have a controlling program for each of the group jobs that controls what happens when the user ends the function of the group job (for example, the update program), it could also test the shutdown switch and do a return. This will end the group job and return control to the previous active group job.
- The Attention key handling program can use the CHKRCDLCK command to determine if the work station user pressed attention when the application had a record locked for update. In this case, the attention program may send a message instructing the user to complete the operation before using the Attention key.

Performance Considerations

Each group job (active or suspended) requires approximately 1K of dedicated main storage in the machine pool. Thus, the number of group jobs that are allowed to be active is a consideration. However, if main storage is not limited, this may be a good way to gain the benefits of separate process access groups (PAGs). This also allows you to avoid the overhead caused by repetitive opening of files and reestablishing environments, and to avoid excessive interactions to access common functions. For information on the machine pool, see Chapter 5, “Performance Guidelines.”

The effect on the system for a large number of suspended jobs is normally small if the dedicated main storage requirement is not a factor.

When a TFRGRPJOB command runs and a new job must be started, the overhead involved is roughly the same as signing on to the system. When the command is run and the group job is already started, the overhead required is roughly the same as using the transfer to a secondary job option on the System Request menu when the secondary job is already active.

If a group job is to be run with any frequency, it is desirable to prevent it from ending. That is, do not end the program, but issue a TFRGRPJOB command to prevent job starting each time the group job function is needed.

The SETATNPGM command causes the current display to be saved when the Attention key is pressed, and to be restored when the Attention key handling program ends. This is roughly the same as using of the System Request menu and has a more noticeable effect on remote work stations.

The controls on the number of jobs active in the system (the MAXJOBS parameter on the CRTSBSD command) are not affected by the number of group jobs active at any time. However, all system values that control the creation of job structures (QACTJOB and QADLACTJ, and QTOTJOB and QADLTOTJ) are affected; these values may need to be increased to allow for the addition of group jobs.

Chapter 4. Examples of Changing Work Management Objects

Once you understand how work management operates using shipped objects and are familiar with some of the CL commands, you should be able to start using work management functions to tailor the system to your operating needs. This chapter contains examples of some of the changes you may want to make to the work management objects supplied by IBM such as:

- Change security level of system to require passwords.
- Provide for and control a completion message for each batch job.
- Use job queues to do long-running functions in a batch environment.
- Add more job queues.
- Create your own interactive or batch subsystems.

Note: If you are currently using QBASE as the controlling subsystem and are considering tailoring your system operations by making changes to the subsystem description such as the ones described here, you should consider changing to use the QCTL controlling subsystem. It is easier to tailor with QCTL because the system activities are divided into separate subsystems that can be ended so you can make changes without doing an IPL. Most changes cannot be made to QBASE without an IPL.

Controlling User Sign-Ons

When a work station user signs on, the following actions occur:

- The routing table is searched for a match for the routing data, which comes from the job description.
- If QSYS/QCMD is the program specified in the routing entry that matched the routing data, the user profile is checked for an initial program. If an initial program is specified, that program is called. If control returns from the initial program, a check is made for an initial menu. If one is specified in the user profile, it is displayed.
- If QSYS/QCMD is not the program specified in the routing entry, the program that is specified is called.

Therefore, there are ways in which you can control what is displayed when a work station user signs on:

- You can create an initial program or initial menu that is specified in a particular user profile and specify QSYS/QCMD in the routing entry. Whenever the user signs on with this user profile, the same information is displayed regardless of which work station he uses.
- You can specify an initial program or an initial menu on the Sign On display if your user profile allows this.
- You can change the work station entry to specify a job description that has routing data to route to a particular routing entry and call a specific program so the routing operates the same regardless of which user signs on at the work station.
- You can use request data in a job description to call a program. In this case, if the work station user has an initial program associated with his user profile, that

initial program is called. However, if there is no initial program associated with the user profile, the request data is processed and the specified program is called.

Using an Initial Program in a User Profile

To specify an initial program in a user profile, you use the Create User Profile (CRTUSRPRF) command:

```
CRTUSRPRF USRPRF(WSUSERS) INLPGM(USRMNUPGM) INLMNU(*SIGNOFF)
```

QSYS/QCMD should also be specified in the routing entry. The initial menu and initial program are called only when QSYS/QCMD is specified in the routing entry. If you route to a different program, the system will not call them. (You can call them with your own program. Use the RTVUSRPRF command to determine what they are.) Whenever someone signs on as WSUSERS, the user profile is checked for an initial program, and the program USRMNUPGM is called unless overridden on the Sign On display. This program can display a menu that allows work station users to select the application they want to run.

If you do not end the initial program, the program continues to run until the subsystem is ended or the job is ended.

For information about ending an initial program, see "Interactive Job Considerations" on page 2-30.

Using an Initial Menu in a User Profile

To specify an initial menu in a user profile, you use the Create User Profile (CRTUSRPRF) command:

```
CRTUSRPRF USRPRF(WSUSERS) INLMNU(MAIN) INLPGM(SETUP)
```

QSYS/QCMD should also be specified in the routing entry. The initial menu and initial program are called only when QSYS/QCMD is specified in the routing entry. If you route to a different program, the system will not call them. (You can call them with your own program. Use the RTVUSRPRF command to determine what they are.) Whenever someone signs on as WSUSERS, the user profile is checked for an initial program, and the program SETUP is called unless overridden on the Sign On display when the program setup is done. The initial menu MAIN is displayed (unless overridden on the Sign On display). The initial menu will continue to be displayed until you sign off.

Using Routing Data

When you use routing data to determine a controlling program for a work station, you need to create a job description that specifies the routing data, add a work station entry to the subsystem description, and add a routing entry to the subsystem description. (In order to change the subsystem description, the subsystem must be inactive.)

```
CRTJOBD JOBD(DSP02JOBD) USER(*RQD) RTGDTA('DSP02')
ADDWSE SBSD(QINTER) WRKSTN(DSP02) JOBD(DSP02JOBD)
ADDRTGE SBSD(QINTER) SEQNBR(100) CMPVAL('DSP02') PGM(DSP02MNU) +
        POOLID(2)
```

Whenever anyone signs on at work station DSP02, the system retrieves the routing data from the associated job description (DSP02JOBD) and compares it with the compare values specified in the routing entries. Because the routing data matches the compare value in the routing entry that was added to the subsystem description for QINTER, the program DSP02MENU is called.

Using Request Data

When you use request data to determine a controlling program for a work station, you need to create a job description that specifies the request data, add a work station entry to the subsystem description specifying this job description, and create a user profile that specifies the password to be used.

```
CRTJOBD JOBD(DSP03JOB) USER(*RQD) RQSDTA('CALL DSP03MENU') RTGDTA(QCMDI)
ADDWSE SBSD(QINTER) WRKSTN(DSP03) JOBD(DSP03JOBD)
CRTUSRPRF USRPRF(DSP03USER) INLPGM(*NONE)
```

If someone signs on at work station DSP03 with user name DSP03USER, the program DSP03MENU is called. However, if someone signs on with user name WSUSERS, the program USERMENU (as described under "Using an Initial Program in a User Profile" on page 4-2) is called because it is specified in the user profile. If DSP03USER is used to sign on at another work station that does not have a special work station entry set up, the initial menu is displayed.

Using Double-Byte Request Data

You can use double-byte data as part of the request data entered with the following commands:

- Reroute Job (RRTJOB)
- Transfer Job (TFRJOB)
- Transfer Batch Job (TFRBCHJOB)

The number of double-byte characters that you can include in request data, including shift control characters, is half the allowed number of alphameric characters.

Note: Do not use double-byte data as request data (RQSDTA) or routing data (RTGDTA) parameters on the Batch Job (BCHJOB) and Submit Job (SBMJOB) commands because the system might not properly process the data.

Preventing Request End or Sign-Off

Normally, a work station user can end a request or sign off by requesting the System Request menu and selecting the appropriate option. This could cause a problem if the work station user ended a request or signed off in the middle of a multiple operation transaction.

You can prevent access to the System Request menu (*Security Concepts and Planning* contains information on how to do this). You can also allow access to the System Request menu, but prevent the use of certain commands. For example, the security officer can revoke public authority to the End Request (ENDRQS) and SIGNOFF commands and give it only to those users that are not running sensitive applications. In order for the work station user to sign off, he must call a CL program that issues the SIGNOFF command. This program must be owned by someone with authority to the command and must be created with the value USRPRF(*OWNER) specified on the Create CL Program (CRTCLPGM) command.

If the work station user's authority to the ENDRQS and SIGNOFF commands is revoked and he attempts to select these options from the System Request menu, he receives a message that he does not have authority to the commands.

Changing the Start-Up (QSTRUP) Program

The autostart job in the controlling subsystem transfers control to the program specified in the system value QSTRUPPGM. You can can tailor this program.

You can create your own program and change the QSTRUPPGM system value to that program name. Or, you can use the shipped program QSTRUP in QSYS as a base to create your own program. To do this, you would:

1. Retrieve the source of the shipped program using the RTVCLSRC command.
2. Change the program.
3. Create the program using the CRTCLPGM command, putting it into your own library.
4. Change the system value QSTRUPPGM to the program name and library you specified on the CRTCLPGM command.

Creating a Programmer Output Queue

You may want to create a special output queue for the programmer so his spooled output is always sent to that queue. This allows the programmer to use the Work with Spooled File (WRKSPLF) command or the source entry utility (SEU) to display the spooled output at his work station and then selectively print or delete the spooled output.

First, you use the Create Output Queue (CRTOUTQ) command to create the output queue and the Grant Object Authority (GRTOBJAUT) command to give the programmer authority to the output queue:

```
CRTOUTQ OUTQ(PGMR) AUT(*NONE) AUTCHK(*DTARIGHTS)
GRTOBJAUT OBJ(PGMR) OBJTYPE(*OUTQ) USER(QPGMR) AUT(*USE)
```

You could change the OUTQ parameter in your user profile, or you could do one of the following:

> For interactive jobs, you can change the initial program for the programmer so the new initial program specifies this output queue:
>
> ```
> PGM
> CHGLIBL LIBL(TESTLIB QGPL QTEMP)
> CHGJOB OUTQ(PGMR)
> TFRCTL QSYS/QPGMMENU
> ENDPGM
> ```
>
> Other functions, such as the logging level for messages, could also be set in this initial program. The libraries specified on the Change Library List (CHGLIBL) command must exist on the system. Generally, the first library specified contains the basic programmer objects such as source files and a job description.
>
> Another alternative is to assign a unique job description to the user profile that contains the desired parameters.
>
> For batch jobs, you create a job description with the Create Job Description (CRTJOBD) command and specify the QGPL/PGMR output queue on the OUTQ parameter.

Adding a Second Job Queue

This example assumes that the system has already been changed to use the QBATCH subsystem for batch work.

The QBATCH subsystem is shipped with a single job queue for submitting batch jobs: QBATCH. A second job queue can be created and added to the subsystem.

To create the job queue, enter the following command:

```
CRTJOBQ   JOBQ(QGPL/QBATCH2) +
          TEXT('Second job queue for QBATCH')
```

To add the job queue to the QBATCH subsystem when the subsystem is inactive, enter the following command:

```
ADDJOBQE  SBSD(QGPL/QBATCH) +
          JOBQ(QGPL/QBATCH2) +
          SEQNBR(20)
```

See "Subsystem Descriptions" on page 2-3 for information on how jobs are selected from multiple job queues in a single subsystem.

Allowing More Batch Jobs to Run in QBATCH at a Time

The QBASE subsystem is shipped with a job queue entry for the QBATCH job queue that only allows one batch job at a time to be running. If you want to allow more batch jobs from that job queue to be running at the same time you would use the Change Job Queue Entry (CHGJOBQE) command. The following command would allow two batch jobs from the QBATCH job queue to be running at the same time in the QBASE subsystem. (This command can be issued at any time and will take effect immediately.)

```
CHGJOBQE SBSD(QBASE) JOBQ(QBATCH) MAXACT(2)
```

Sending a Completion Message for Each Batch Job

You can use the MSGQ parameter on the BCHJOB or Submit Job (SBMJOB) command to specify the name of the message queue to which a completion message is to be sent when the job has completed (message CPF1241 indicates a normal completion and CPF1240 indicates an abnormal completion). For example, to send a completion message to the QSYSOPR message queue, you could use the following BCHJOB command:

```
//BCHJOB JOB(PAYROLL) JOBD(PAYROLL) MSGQ(QSYSOPR)
```

Instead of sending the completion messages for all jobs to the QSYSOPR message queue, you may want to send the completion messages for some jobs to the message queue of the submitting work station and for other jobs to a message queue that is periodically displayed or printed.

Another alternative is to send all the completion messages to a user message queue. You can then use a program to receive the messages from this queue and to resend them if necessary. For example, if a job failed for a user who is not a programmer or the system operator, the program could resend the message to both the work station user and the system operator.

Setting Up an Unattended Environment

To set up an unattended environment and prevent unauthorized users from using the console while the system is unattended, have the system operator change the QSYSOPR message queue to default delivery and sign off. The sign-off prevents unauthorized users from using the console. The following command changes the QSYSOPR message queue:

```
CHGMSGQ MSGQ(QSYSOPR) DLVRY(*DFT)
```

Setting Up an Unattended Nighttime Environment

You may have jobs in your installation that can be run in an unattended nighttime environment. These jobs could be submitted throughout the day to be processed at night. To set up this environment, you could create a special subsystem for the nighttime work. The subsystem description and job queue could be defined with the following commands:

```
CRTSBSD    SBSD(NIGHTQ) POOLS((1 *BASE)) TEXT(' +
           Nighttime jobs')
CRTJOBQ    JOBQ(NIGHTQ) TEXT('Nighttime job queue')
ADDJOBQE   SBSD(NIGHTQ) JOBQ(NIGHTQ) MAXACT(1)
ADDRTGE    SBSD(NIGHTQ) SEQNBR(10) +
           CMPVAL(*ANY) PGM(QSYS/QCMD) CLS(QGPL/QBATCH)
```

You then create a job description using the Create Job Description (CRTJOBD) command that specifies the job queue QGPL/NIGHTQ on the JOBQ parameter.

When ready to start the NIGHTQ subsystem, the system operator:

1. Ends all active subsystems except the controlling subsystem using the End Subsystem (ENDSBS) command.
2. Starts the NIGHTQ subsystem with the Start Subsystem (STRSBS) command.
3. Changes the QSYS/QSYSOPR message queue to place it in default mode so any messages requiring a response receive a default reply.
4. Submits a job with request data of a Power Down System (PWRDWNSYS) command and a job priority of 9 to the NIGHTQ job queue. This priority places the job at the end of the job queue so the system will be powered down when all the jobs in the job queue have been processed.
5. Signs off.

The STRSBS command, the Change Message Queue (CHGMSGQ) command, and the PWRDWNSYS command could be placed in a CL program to simplify the operator's job. The CL program could be:

```
PGM
CHGMSGQ   QSYSOPR DLVRY(*DFT)
STRSBS    NIGHTQ
SBMJOB    JOBD(QBATCH) JOBQ(NIGHTQ) JOBPTY(9) +
          CMD(PWRDWNSYS *IMMED)
SNDPGMMSG MSG('NIGHTQ subsystem started and power down +
          job submitted')
ENDPGM
```

This solution assumes that only one job is run at a time from the NIGHTQ job queue and that no other jobs are running while the nighttime subsystem is running. If this is not desirable, the procedure should be changed as in the following example so that two jobs run at once and when they are completed, the system powers down.

The following CL program could be used to start the NIGHTQ subsystem with two batch jobs running and to ensure that all jobs have finished before the system is powered down. The job queue entry shown in the previous example should be changed to meet your requirements.

```
PGM
CHGMSGQ QSYSOPR DLVRY(*DFT)
CHGSBSD NIGHTQ MAXJOBS(2) /* Reset to 2 */
STRSBS NIGHTQ
SBMJOB JOBD(QBATCH) JOBQ(NIGHTQ) JOBPTY(9) CMD(CHGSBSD NIGHTQ +
            MAXJOBS(1)) JOB(CHGMAXJOBS)
SBMJOB JOBD(QBATCH) JOBQ(NIGHTQ) JOBPTY(9) +
            CMD(PWRDWNSYS *IMMED) JOB(POWERDOWN)
SNDPGMMSG MSG('NIGHTQ subsystem started and power down job +
            submitted')
ENDPGM
```

When the subsystem is started, the MAXJOBS value is set to 2. The first job submitted (CHGMAXJOBS) then sets the MAXJOBS value to 1 just before the power down job is run. Because the priority of both the CHGMAXJOBS and POWERDOWN jobs is 9, neither job runs until all higher priority jobs in the queue (0 being the highest priority) are run.

Calling a Special Recovery Program

To call a special recovery program if the IPL senses that the previous system ending was abnormal, you can add an autostart job entry to the subsystem description for the controlling subsystem. This program checks the system value QABNORMSW. For a normal system ending, the value of QABNORMSW is '0', and for an abnormal system ending the value of QABNORMSW is '1'.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
/* SPCRECOV - Autostart program to call special recovery program */
             PGM
             DCL        &QABNORMSW *CHAR LEN(1)
             RTVSYSVAL  SYSVAL(QABNORMSW) RTNVAR(&QABNORMSW)
             IF         (&QABNORMSW *EQ '1') DO /* Recover */
             SNDPGMMSG  MSG('Recovery program in operation-do not +
                          start subsystems until notified') +
                          TOMSGQ(QSYSOPR)
             CALL       RECOVERY
             SNDPGMMSG  MSG('Recovery complete-jobs may be started') +
                          TOMSGQ(QSYSOPR)
             ENDDO      /* Recover */
             ENDPGM
```

An alternative would be to drop the messages and start up other subsystems when your recovery function is complete.

Preventing Sign On Display from QINTER

This example assumes that the system has been changed to use the QINTER subsystem for interactive work.

If you do not want the Sign On display shown at all work stations when you start the QINTER subsystem, you can do the following:

Change the work station entries in the QGPL/QINTER subsystem description to specify the work stations by name (WRKSTN), rather than by type (WRKSTNTYPE). For the work stations for which you want the Sign On display to appear, specify AT(*SIGNON) in the work station entry. For the work stations for which you do not want the Sign On display to appear, specify AT(*ENTER). This allows interactive jobs to be transferred to work stations with AT(*ENTER) specified in the QGPL/QINTER subsystem.

Performing a Long-Running Function from a Work Station

To perform a long-running function (such as save/restore) from a work station without tying it up, the system operator can submit the job to a job queue. The controlling subsystem description QGPL/QCTL or QGPL/QBASE, which is supplied by IBM, has a job queue QSYS/QCTL that can be used for this purpose. If you created your own controlling subsystem, you should refer to the job queue for that subsystem. The system operator can submit the commands from the system operator menu. The following is an example of submitting a long-running command:

```
SBMJOB    JOB(SAVELIBX) JOBD(QBATCH) JOBQ(QSYS/QCTL) +
          CMD(SAVLIB LIBX CLEAR(*YES))
```

Controlling a Group of Work Stations

You may want to control a group of work stations separately. For example, if you have a group of work station users who are using the same application, you may want that group of users to sign off to perform some special application function or to send them a message. You can:

- Create a separate subsystem for the group of work station users. A separate subsystem will allow a convenient start and shutdown, but it will not simplify the job of sending messages to the group.
- Create a command or a CL program that can be used to send a message to a group of work stations. However, this approach does not assist in controlling sign-on or sign-off. See the *CL Programmer's Guide*, for an example of a CL program that can be used to send a message to a group of work stations.

Changing Priority and Time Slice

You can create a program that changes your priority and time slice for the duration of the command running and then resets the two values to what they were previously. For example, it may be desirable to have a special program that can be used in an emergency situation where the response time of command running is critical. To do this, you would enter the following program to improve the performance of the WRKACTJOB command:

```
PGM
DCL &RUNPTY *DEC LEN(2 0)
DCL &TIMESLICE  *DEC LEN(7 0)
RTVJOBA  RUNPTY(&RUNPTY) TIMESLICE(&TIMESLICE)
CHGJOB RUNPTY(1) TIMESLICE(20000)
WRKACTJOB
CHGJOB RUNPTY(&RUNPTY) TIMESLICE(&TIMESLICE)
ENDPGM
```

You may want to create a command for this function to allow a simple means of running this program.

If the WRKACTJOB command is being used to determine an immediate performance situation on the system, you should consider specifying the command as:

```
WRKACTJOB CPUPCTLMT(2) RESET(*YES)
```

When the display first appears, there will be no active jobs meeting the requirements because the active job statistics have been reset. If the Refresh key is pressed, the display will then show all jobs that are using more than 2% of the processing unit resource since the command was run.

Mixing Double-Byte and Alphameric Display Stations

If you use both double-byte and alphameric devices with your system, you can create an initial program to have double-byte versions of user-written alphameric messages and user-designed displays shown at double-byte work stations. The following procedure describes how to show the double-byte versions of messages and displays at double-byte work stations:

1. Create a library for storing the double-byte versions of messages and display files. For example, to create a library named DBCSLIB, enter the following command:

   ```
   CRTLIB LIB(DBCSLIB) TEXT('Double-byte version of objects')
   ```

2. Put a copy of the messages and display files into the library you just created.
3. Translate the information in the copied file from alphameric information to double-byte information. For example, change message text from alphameric to double-byte.

   Do not change the file name.

4. Create a job description to be used for the double-byte devices. This job description should be similar to that used for alphameric work stations. However, the job description has different routing data. For example, to create the job description DBCSJOBD, enter the following command:

   ```
   CRTJOBD JOBD(DBCSJOBD) RTGDTA(DBCSWS) LOG(4 0 *SECLVL) +
   TEXT('Double-byte work station job description')
   ```

5. Create a copy of the subsystem description. This description will be changed for use as a subsystem description for DBCS work stations. For example, to create a copy of the subsystem description named QINTER as the subsystem description DBCSSBSD, enter the following command:

   ```
   CRTDUPOBJ OBJ(QINTER) FROMLIB(QGPL) OBJTYPE(*SBSD) +
   NEWOBJ(DBCSSBSD)
   ```

6. Change the work station entry for the 5555 Display to specify the job description you just created. For example, to specify a job description named DBCSJOBD, enter the following command:

   ```
   CHGWSE SBSD(DBCSSBSD) WRKSTNTYPE(5555) JOBD(DBCSJOBD)
   ```

7. Add a routing entry that matches the routing data of the double-byte job description. For example:

   ```
   ADDRTGE SBSD(DBCSSBSD) SEQNBR(15) CMPVAL(DBCSWS) +
   PGM(DBCSPGM)
   ```

8. Create a CL program named DBCSPGM, similar to the one shown below, that performs the following functions:

   a. Inserts the double-byte library at the start of the system library list before transferring to the system module QCMD.

   b. Overrides the default printer file so that the file is double-byte. When the default printer file is double-byte, the system properly prints double-byte output printed as a result of pressing the Print key.

   ```
   PGM
      CHGSYSLIBL LIB(DBCSLIB)
      OVRPRTF FILE(QSYS/QSYSPRT) IGCDTA(*YES)
      TFRCTL QSYS/QCMD
   ENDPGM
   ```

When you use the subsystem description DBCSSBSD, the double-byte versions of user-written messages and user-designed displays that are stored in the library DBCSLIB are shown at double-byte work stations. The alphameric versions of these messages and displays continue to be shown at alphameric work stations.

Time-Dependent Scheduling

Some applications require that functions be performed at a specified time. This is referred to as time-dependent scheduling.

Although the system does not support this type of function directly, existing functions can be combined to perform time-dependent scheduling.

A single request-processing job must be active at all times to handle all of the time-dependent requests sent by other jobs. These requester jobs can send one or more requests, which are then run or released by the single request-processing job when they are due.

The following sample commands illustrate two different uses for time-dependent scheduling.

- Send Time-Dependent Message (SNDTIMMSG): Runs a SNDMSG command at the requested time.
- Submit Time-Dependent Job (SBMTIMJOB): Runs a SBMJOB command with HOLD(*YES) and is released at the requested time.

SNDTIMMSG Sample Command

The SNDTIMMSG (Send Time-Dependent Message) sample command would be used for simple functions where you want to send a message to yourself or a work station user at a specific time as a reminder, for example, to attend a meeting or make a telephone call. It is intended for personal rather than program action.

This sample command would have the following parameters:

MSG	The message text. Up to 100 characters can be sent.
HOUR	The hour the message should be sent. The 24-hour clock is used (2 p.m. is entered as 14). A special value of *NOW is allowed so that the message can be sent immediately. This allows users to have a single command to do both the delayed time and immediate function. If *NOW is specified and a different date is entered, the current time is used for the specified date.
MIN	The minute the message should be sent. The default is 00.
DATE	The date on which the message should be sent. The default is *TODAY, which sends the message the same day. Other special values of *DAY1, *DAY2 through *DAY14 allow for sending the message any time from 1 through 14 days from today. An actual date may also be entered in the system format (for example, MMDDYY).
TOMSGQ	The qualified name of the message queue that is to receive the message. The default is *USER, which means to send it back to the requesting user's message queue.

The user name of the requesting user is concatenated to the beginning of the message. If user JONES enters:

```
MSG   Will you meet me for lunch?
```

it appears as the following message:

```
From Jones --- Will you meet me for lunch?
```

If you are sending messages that contain apostrophes, a special form may be required. If the SNDTIMMSG command is entered by using the prompter, the message should be entered with single apostrophes, such as the following:

```
MSG   Don't forget your meeting tomorrow
```

If the command is entered in a CL program, or if the prompter is not used to enter the SNDTIMMSG sample command, the message should be written with surrounding apostrophes for the MSG parameter and two apostrophes for each one desired in the text of the message, such as the following:

```
MSG('Don"t forget your meeting tomorrow')
```

If the message text is in a CL variable, there should be only one apostrophe in the message for each apostrophe desired.

The SNDTIMMSG sample command rejects any attempt to send a message for a date and time that precedes the current system date and time. This minimizes errors occurring when the date or time is incorrectly specified.

SBMTIMJOB Sample Command

The SBMTIMJOB (Submit Time-Dependent Job) sample command allows any job to be submitted to a job queue with HOLD(*YES). A RLSJOB command is then added to the TIMFILP data base file. The RLSJOB command is run when it is due. The job in the held status acts as a normal job on the job queue and does not require an activity level until it is released and is running.

Note: The job is not run at the requested time; rather it is released at that time. All held jobs appear last on the job queue. A job is released according to the sequence of its job number in relation to the sequence number of all other jobs of the same priority level. If it is critical to perform certain functions at the requested time, then a unique job queue could be created to handle these requests.

This SBMTIMJOB sample command supports the following parameters:

RQSDTA	The request data to be submitted. Up to 256 bytes may be entered.
HOUR	The hour the job should be released. The 24-hour clock is used (2 p.m. is entered as 14). A special value of *NOW is allowed so that the message can be sent immediately. This approach allows users to have a single command to do both the delayed time and immediate function. If *NOW is specified and a different date is entered, the current time is used for the specified date.
MIN	The minute the job should be released. The default is 00.
DATE	The date on which the job should be released. The default is *TODAY, which releases the job the same day. Other special values (*DAY1, *DAY2 through *DAY14) allow for releasing the job any time from 1 through 14 days from today. An actual date may also be entered in the system format (for example, MMDDYY).
JOB	The job name to be assigned to the submitted job. The default is the job description name.
JOBD	A fully qualified job description name to be used for the submitted job. The default is QUSER, which uses the job description specified in the requesting user's profile.
WRKSBMJOB	A *YES/*NO value that has the same definition as the parameter on the SBMJOB command. The default is *NO, which may be used to prevent the submitted job from being displayed by the WRKSBMJOB command.

If the prompter is used to enter the SBMTIMJOB sample command, the RQSDTA parameter may be entered as ?XXXXX on the prompt display. The other parameters for the SBMTIMJOB command should be filled in before pressing the Enter key. The next display is the prompt for the XXXXX command. For example, assume that the CPYF command should be submitted at a specific time. The user would request the prompter for the SBMTIMJOB command and fill in the appropriate parameters. The RQSDTA parameter should be entered as ?CPYF. The next display is the prompt for the CPYF command. This simplifies entering the command to be run.

It is not necessary to use prompting for the SBMTIMJOB command in order to use the ? function to prompt for the RQSDTA parameter. If the command is entered as follows, the prompter would be called for the CPYF command.

```
SBMTIMJOB RQSDTA(?CPYF) HOUR(16)
```

Note: The special value of *NOW in the HOUR parameter allows the command to act like the SBMJOB command with prompting for the request data. That is, the user can submit a command to run immediately and still receive prompting for the submitted command, such as:

```
SBMTIMJOB RQSDTA(?CPYF) HOUR(*NOW)
```

If you are submitting commands that require apostrophes, a special form may be required. If the SBMTIMJOB command is entered with prompting, then the command to be submitted must be entered with single apostrophes, such as:

```
CALL PGMA PARM('ABC' 'XYZ')
```

If the command is entered in a CL program, the RQSDTA parameter should be written with surrounding apostrophes and two apostrophes for each one desired in the run command, such as:

```
RQSDTA('CALL PGMA PARM(''ABC'' ''XYZ'')')
```

At the completion of the submitted job, the job completion message is sent to the originating work station.

The SBMTIMJOB sample command rejects any attempt to submit a job for a date and time that precedes the current system date and time. This minimizes errors occurring when the date or time is incorrectly specified.

The WRKSBMJOB(*NO) option allows a standard function such as an initial program to submit a job for later running. The job will not appear on the WRKSBMJOB display with directly submitted jobs.

Starting the Request-Processing Job

The request-processing job would normally be started at the beginning of the day, possibly as an autostart job associated with a particular subsystem. In this example, the program to be called is TIMPGMC, a CL program that will call the RPG program TIMPGM when needed.

The request-processing job runs as an interactive job for most of its functions; you may choose to place it in the interactive subsystem. The program remains active until you end it (see the sample command ENDTIMSCH later in this chapter). It is active in main storage only when new requests arrive or a time-out occurs. If the job is placed in a batch subsystem, you will want to be sure that the maximum number of jobs specified will allow the request-processing job as well as your other jobs to be started. To submit the job to the QINTER job queue, the following command could be used:

```
SBMJOB JOB(TIMESCHED) JOBD(XXXX) RQSDTA('CALL TIMPGMC') +
       JOBQ(QINTER) INQMSGRPY(*SYSRPYL)
```

See "Security Considerations" on page 4-16 later in this chapter about the required *JOBCTL special value.

The INQMSGRPY parameter is specified to use the system reply list, which ensures problem analysis help if an inquiry message occurs (assuming that the response to any inquiry message will provide a program dump).

When the program is started, it clears out any requests that have been run from the last time the program was started. When a request is run, the record in TIMFILP is updated, reflecting, for problem analysis purposes, what was entered, who entered it, and when it was run. When the program is started, the previous requests that ran are deleted by copying the file into a file in QTEMP and then copying it back again and omitting the completed requests.

For an audit trail, you could either journal the file, change the TIMPGM program to print the requests that ran before deleting them, or cause the records to be printed with the CPYF command before deleting them.

When the request-processing program starts, there may be overdue requests in the file that have been entered and already come due before the request-processing program is started. If this occurs, these requests are run immediately.

The request-processing job may be ended by using the End Job (ENDJOB) command or by the sample command End Time Scheduling (ENDTIMSCH). There are no parameters for this command and it is entered as:

```
ENDTIMSCH
```

Security Considerations

You may authorize the commands and the programs as required.

The command entered on the SBMTIMJOB sample command is submitted for processing with the user profile of the requester, because the SBMJOB command defaults to USER(*CURRENT). It is the request-processing job that releases the requester job. To release another user's job, the request-processing job must be started by a user who has special authority *JOBCTL.

The running of the SNDTIMMSG command causes the running of the SNDMSG command to occur in the request-processing job, under the user profile who started the request-processing job. If the messages are sent only to work station message queues, there should be minimal security considerations. As in a normal environment, you may wish to control who can send messages to certain user-created message queues.

Recovery Considerations

If an error occurs on a command so that it must be run again, the command would need to be reentered.

If an abnormal system ending occurs, the normal procedure for a full recovery would be to start the request-processing job again. However, in time-dependent scheduling there is an interval between the time that a job is placed on the job queue and the time that a record is added to the TIMFILP file or between the time that a record in the TIMFILP is coded as sent and the time that the command is run. This interval is small enough so that in most cases an unsynchronized condition does not occur. However, following an abnormal ending, you may want to check manually to ensure that the TIMFILP and the jobs on the job queue are synchronized.

Creating the Objects

The sample objects should be placed in a library such as QGPL where all users can access them by use of their library list. If QGPL is not used, you will need to change where you specifically refer to QGPL in the sample code.

In the following example, the data base file TIMFILP is used to hold the requests to be processed. The SNDTIMMSG and SBMTIMJOB functions add records to this file. A request-processing job remains continuously active waiting to process requests.

Adding a record to a data base file will not cause the request-processing program to activate. Therefore, a data queue is used to allow the request-processing program to monitor for new requests and to wait for the timeout to run the due requests. Data queues are discussed in the *CL Programmer's Guide* and are called by the QSNDDTAQ and QRCVDTAQ OS/400-supplied programs.

TIMDTAQ Data Queue

The TIMDTAQ data queue is created automatically in QGPL as part of the TIMPGMC program. The data queue will be deleted and re-created each time the TIMPGMC program is run.

```
CRTMSGQ XXXX/TIMMSGQ TEXT('Time-dependent scheduling queue')
```

TIMFILP Physical File

The following is the DDS for the TIMFILP; it should be created with the CRTPF command.

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00      A          R TIMFILR                    TEXT('Time Rcd Format')
  2.00      A            DLTCOD         1           COLHDG('Delete' 'Code')
  3.00      A                                       TEXT('Delete Code - +
  4.00      A                                       Blank = Ready, +
  5.00      A                                       X = Complete')
  6.00      A            RQSDAT         6           COLHDG('Request' 'Date')
  7.00      A            RQSTIM         6           COLHDG('Request' 'Time')
  8.00      A            NBRSEC         5  0        COLHDG('Number of' +
  9.00      A                                        'Seconds' 'For Request')
 10.00      A            ENTDAT         6  0        COLHDG('Entry' 'Date')
 11.00      A            ENTTIM         6  0        COLHDG('Entry' 'Time')
 12.00      A            JOB           10           COLHDG('Job name')
 13.00      A            USER          10           COLHDG('User name')
 14.00      A            JOBNBR         6           COLHDG('Job nbr')
 15.00      A            SNTDAT         6  0        COLHDG('Sent' 'Date')
 16.00      A            SNTTIM         6  0        COLHDG('Sent' 'Time')
 17.00      A            CMD          256           COLHDG('Command')
 18.00      A            SBM100       100           TEXT('1st 100 bytes +
 19.00      A                                       of rqs dta submitted or +
 20.00      A                                       SNDTIMMSG')
 21.00      A          K DLTCOD
 22.00      A          K RQSDAT
 23.00      A          K NBRSEC
```

The field information for ENTDAT, ENTTIM, JOB, USER, JOBNBR, SNTDAT, SNTTIM, and SBM100 is only for problem analysis and audit trail purposes. The SBM100 field contains the first 100 bytes of the RQSDTA parameter of the SBMTIMJOB command. The RQSTIM field is for readability purposes only. The data is converted from the RQSTIM field to the NBRSEC field (the time in number of seconds) for ease of processing.

The file is kept in a key sequence of DLTCOD, DATE, NBRSEC. The DLTCOD is blank for a normal record and X if the request has been run. The access path definition ensures that the first record in the file is the first request to be run. After it is run, it is updated with an X and becomes the last record in the access path. As mentioned, the starting of the request-processing program removes the records with a delete code of X.

SNDTIMMSG Command and Processing Program

The following is the source for the SNDTIMMSG command (you may wish to change this command to allow such functions as a specific time or a number-of-seconds delay):

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
       /*PARMS PGM(SNDTIMMSGC)                                             */
       /*                                                                  */
       /*  Send time message sends a message at a specified time which   */
       /*    can be now, sometime today or at a specified date.            */
       /*                                                                  */
       /*  The CPP is SNDITMSMGC. Associated commands - SBMTIMJOB/ENDTIMSCH*/
       /*                                                                  */
       /*  The TIMPGMC and TIMADD programs must be active.                 */
       /*                                                                  */
                     CMD        PROMPT('Send Time Message')
                     PARM       MSG TYPE(*CHAR) LEN(100) MIN(1) EXPR(*YES) +
                                  PROMPT('Message text:')
                     PARM       HOUR TYPE(*CHAR) LEN(2) MIN(1) +
                                  RANGE(00 23) SPCVAL(('*NOW' '**')) +
                                  PROMPT('Hour as 24 hr clock or *NOW:')
                     PARM       MIN TYPE(*CHAR) LEN(2) DFT(00) +
                                  RANGE(00 59) +
                                  PROMPT('Minute (00-59):')
                     PARM       KWD(DATE) TYPE(*DEC) LEN(6 0) DFT(*TODAY) +
                                  RANGE(000000 999999) SPCVAL((*TODAY 0) +
                                  (*DAY1 -1) (*DAY2 -2) (*DAY3 -3) (*DAY4 -4) +
                                  (*DAY5 -5) (*DAY6 -6) (*DAY7 -7) (*DAY8 -8) +
                                  (*DAY9 -9) (*DAY10 -10) (*DAY11 -11) +
                                  (*DAY12 -12) (*DAY13 -13) (*DAY14 -14)) +
                                  PROMPT('Date (sys fmt) *TODAY *DAYnn:')
                     PARM       KWD(TOMSGQ) TYPE(QUAL1) DFT(*WRKSTN) +
                                  SNGVAL((*WRKSTN)) PROMPT('Message queue +
                                  (*WRKSTN):')
       QUAL1:        QUAL       TYPE(*NAME) LEN(10) EXPR(*YES)
                     QUAL       TYPE(*NAME) LEN(10) DFT(*LIBL) +
                                  SPCVAL((*LIBL)) EXPR(*YES) PROMPT('Library:')
```

The following is the source for the SNDTIMMSGC CL program:

```
*... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
/* SNDTIMMSGC - Send time message - CPP for SNDTIMMSG               */
             PGM        PARM(&MSG &HR &MIN &DATEI &MSGQF)
             DCL        &MSG *CHAR LEN(100)
             DCL        &HR *CHAR LEN(2)
             DCL        &MIN *CHAR LEN(2)
             DCL        &DATEI *DEC LEN(6 0) /* Input date */
             DCL        &TIME *CHAR LEN(6)
             DCL        &MSGO *CHAR LEN(110)
             DCL        &DATEO *CHAR LEN(6) /* Output date */
             DCL        &DATEJA *CHAR LEN(5) /* Julian alpha date */
             DCL        &DATEJN *DEC LEN( 5 0) /* Julian dec date */
             DCL        &CURDAT *CHAR LEN(6) /* Current date */
             DCL        &CURDATJA *CHAR LEN(6) /* Julian cur date */
             DCL        &DAYS *DEC LEN(3 0)
             DCL        &DAYSA *CHAR LEN(3)
             DCL        &YEAR *DEC LEN(2 0)
             DCL        &YEARA *CHAR LEN(2)
             DCL        &LEAP *CHAR LEN(18) VALUE('000408121684889296')
             DCL        &FLDLEN *DEC LEN(5 0) VALUE(2)
             DCL        &X *DEC LEN(3 0) VALUE(-1) /* Index to leap */
             DCL        &ENDYR *DEC LEN(3 0) VALUE(365)
             DCL        &DATEA *CHAR LEN(6)  /* Date in alpha */
             DCL        &CURHR *CHAR LEN(2)
             DCL        &CURMIN *CHAR LEN(2)
             DCL        &MSGQF *CHAR LEN(20) /* Full message queue */
             DCL        &CMD *CHAR LEN(256)
             DCL        &JOB *CHAR LEN(10)
             DCL        &USER *CHAR LEN(10)
             DCL        &JOBNBR *CHAR LEN(6)
             DCL        &NOW *LGL
             DCL        &FROM *DEC LEN(3 0) VALUE(1) /* From index */
             DCL        &TO *DEC LEN(3 0) VALUE(1)   /* To index */
             DCL        &SBM100 *CHAR LEN(100) /* 1st 100 of msg */
             RTVJOBA    JOB(&JOB) USER(&USER) NBR(&JOBNBR)
             IF         (%SST(&MSGQF 1 10) *EQ '*WRKSTN') +
                          CHGVAR &MSGQF (&JOB *CAT 'QSYS')
             CHKOBJ     OBJ(%SST(&MSGQF 1 10).%SST(&MSGQF 11 10)) +
                          OBJTYPE(*MSGQ) /* Does MSGQ exist */
             MONMSG     MSGID(CPF0000) EXEC(SNDPGMMSG MSGID(CPF9898) +
                          MSGF(QCPFMSG) MSGTYPE(*ESCAPE) MSGDTA('+
                          Unable to access message queue ' *CAT +
                          %SST(&MSGQF 1 10) *TCAT '.' *TCAT +
                          %SST(&MSGQF 11 10)))
             RTVSYSVAL  QHOUR RTNVAR(&CURHR)
             RTVSYSVAL  QMINUTE RTNVAR(&CURMIN)
             RTVSYSVAL  QDATE RTNVAR(&CURDAT)  /* Today's date */
             IF         (&HR *EQ '**') DO /* *NOW converted to ** */
             CHGVAR     &HR &CURHR  /* Set to current */
             CHGVAR     &MIN &CURMIN  /* Set to current */
             IF         (&DATEI *EQ 0) CHGVAR &NOW '1' /* Set to on */
             ENDDO      /* *NOW converted to ** */
             CHGVAR     &DATEA &CURDAT  /* Start with today's date */
             IF         (&DATEI *GT 0) CHGVAR &DATEA &DATEI /* Value */
             IF         (&DATEI *LT 0) DO /* *DAYnn function */
```

```
*... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
             CVTDAT     &DATEA TOVAR(&DATEJA) TOFMT(*JUL) TOSEP(*NONE)
             CHGVAR     &DATEJN &DATEJA  /* Change to numeric */
             CHGVAR     &DATEJN (&DATEJN - &DATEI) /* Sub neg nbr */
             CHGVAR     &DATEJA &DATEJN  /* Change to alpha */
                        /* Check for date which is in the next year */
             CHGVAR     &YEARA %SST(&DATEJA 1 2) /* Extract year */
             CHGVAR     &DAYSA %SST(&DATEJA 3 3) /* Extract days */
             CHGVAR     &YEAR &YEARA /* Change to numeric */
             CHGVAR     &DAYS &DAYSA /* Change to numeric */
 LEAPLOOP:   CHGVAR     &X (&X + 2) /* Index to leap year years */
             IF         (&X *LT 19) DO /* &X is less than 19 */
             IF         (%SST(&LEAP &X 2) *EQ &YEARA) CHGVAR &ENDYR 366
             GOTO       LEAPLOOP
             ENDDO      /* &X is less than 19 */
             IF         (&ENDYR *LT &DAYS) DO /* Bump for next year */
             CHGVAR     &DAYS (&DAYS - &ENDYR) /* Subtract end year */
             CHGVAR     &YEAR (&YEAR + 1) /* Increment year */
             CHGVAR     &YEARA &YEAR /* Change to alpha */
             CHGVAR     &DAYSA &DAYS /* Change to alpha */
             CHGVAR     %SST(&DATEJA 1 2) &YEARA /* Move new days */
             CHGVAR     %SST(&DATEJA 3 3) &DAYSA /* Move new year */
             ENDDO      /* Bump for next year */
             CVTDAT     &DATEJA FROMFMT(*JUL) TOVAR(&DATEA) +
                         TOSEP(*NONE)
             ENDDO      /* *DAYnn function */
             CVTDAT     &DATEA TOVAR(&DATEO) TOFMT(*YMD) TOSEP(*NONE)
             MONMSG     MSGID(CPF0000) EXEC(GOTO BADDATE)
             CVTDAT     &CURDAT TOVAR(&CURDATJA) TOFMT(*YMD) +
                         TOSEP(*NONE)
             IF         (&CURDATJA *GT &DATEO) GOTO BADDATE /* Prior */
             IF         (&CURDATJA *EQ &DATEO) DO /* Current date */
             IF         ((&CURHR *GT &HR) *OR ((&CURHR *EQ &HR) *AND +
                          (&CURMIN *GT &MIN))) GOTO BADTIME
             ENDDO      /* Current date */
                        /* Loop to dbl apostrophes in the msg text */
 NEXT:       IF         (%SST(&MSG &FROM 1) *EQ '''') DO /* apos */
             CHGVAR     %SST(&MSGO &TO 2) '''''' /* Double apos */
             CHGVAR     &TO (&TO + 2)  /* Increment TO index by 2 */
             ENDDO      /* Found apostrophe */
             ELSE       DO  /* Not an apostrophe */
             CHGVAR     %SST(&MSGO &TO 1) %SST(&MSG &FROM 1)
             CHGVAR     &TO (&TO + 1)  /* Increment TO index by 1 */
             ENDDO      /* Not an apostrophe */
             CHGVAR     &FROM (&FROM + 1) /* Increment FROM inx by 1 */
             IF         (&FROM *LT 100) GOTO NEXT
                        /* Build command to add to time file */
             CHGVAR     &TIME (&HR *CAT &MIN *CAT '00')
             CHGVAR     &CMD ('SNDMSG MSG(' *CAT '''' *CAT 'From ' +
                          *CAT &USER *TCAT ' --- ' *CAT &MSGO *CAT +
                         '''' *CAT ') TOMSGQ(' *CAT %SST(&MSGQF 1 10) +
                          *TCAT '.' *CAT %SST(&MSGQF 11 10) *CAT ')')
             CHGVAR     &SBM100 &MSG /* 1st 100 bytes of msg */
             CALL       TIMADD PARM(&DATEO &TIME &CMD &JOB &USER +
                          &JOBNBR &SBM100)
             CALL       QSNDDTAQ PARM(TIMDTAQ QGPL &FLDLEN 'AA')
```

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
110.00              IF         &NOW SNDPGMMSG MSG('Message sent immediately')
111.00              ELSE       SNDPGMMSG MSG('Message will be sent after +
112.00                           time delay')
113.00              RETURN
114.00   BADDATE:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
115.00                           MSGDTA('Date is prior to current system +
116.00                           date or it is invalid')
117.00   BADTIME:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
118.00                           MSGDTA('Hour/Minute is less than or +
119.00                           equal to the current system time')
120.00              ENDPGM
```

The program accepts the parameters from the command and determines the message queue, job name, and so forth. It ensures that the message queue exists if a specific one is named. If the special value *NOW was entered on the HOUR parameter, the command definition changed this to **. If this value exists, the current hour and minute are used.

The &DATEI variable contains the DATE parameter as passed by the command. If a value is entered, it is used as a specific date. If *TODAY (the default) is entered, a zero is passed from the command, and the current date is used. If the *DAYnn function is used the value passed from the command is a negative value in the form of -1 to -14. This allows the program to distinguish between an actual date and the *DAYNN function. If &DATEI is negative, the current date is converted to a Julian format, and the negative value is subtracted from the current date. This has the effect of adding the number of days requested. The actual date to be used is checked to see if it is valid. If the day requested is the current day, the time must be equal to or greater than the current time.

The message is moved to a separate area one character at a time so the program can check every character entered for an apostrophe. If any are found, two apostrophes are inserted so that the command will be correctly interpreted when it is run. The program builds the SNDMSG command to be run and passes it and control parameters to the TIMADD program, which adds the request to the TIMFILP file. The program sends an entry message to the TIMDTAQ data queue to activate the request-processing program. This is done by a call to the QSNDDTAQ IBM-supplied program.

The following is the source for the TIMADD RPG command:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00      F* TIMADD - Adds records to the time file and the special 9s rcd
  2.00      FTIMFILP IF  E           K        DISK                      A
  3.00      IRQSTIM      DS
  4.00      I                                        1   20HR
  5.00      I                                        3   40MIN
  6.00      I                                        5   60SEC
  7.00      C           *ENTRY    PLIST
  8.00      C                     PARM           RQSDAT
  9.00      C                     PARM           RQSTIM
 10.00      C                     PARM           CMD
 11.00      C                     PARM           JOB
 12.00      C                     PARM           USER
 13.00      C                     PARM           JOBNBR
 14.00      C                     PARM           SBM100
 15.00      F* Normal add case
 16.00      C           '999999'  IFNE RQSDAT                     Normal
 17.00      C           3600      MULT HR        NBRSEC           Hrs to secs
 18.00      C           60        MULT MIN       SAV5N   50       Min to secs
 19.00      C                     ADD  SAV5N     NBRSEC
 20.00      C                     ADD  SEC       NBRSEC           Secs in day
 21.00      C                     MOVE UDATE     ENTDAT           Entry date
 22.00      C                     TIME           ENTTIM           Entry time
 23.00      C                     WRITETIMFILR
 24.00      C                     END                             Normal
 25.00      F* Special 9s record - ensure 9s record exists in file
 26.00      C           '999999'  IFEQ RQSDAT                     9s
 27.00      C           KEY       KLIST
 28.00      C                     KFLD           DLTCOD
 29.00      C                     KFLD           RQSDAT
 30.00      C           KEY       SETLLTIMFILR                  20 IF =
 31.00      C  N20                WRITETIMFILR                    Add
 32.00      C                     END                             9s
 33.00      C                     SETON                     LR
 34.00      C                     RETRN
```

The TIMADD program is used for two different functions. Its normal purpose is to add a new request to the TIMFILP file. It also adds the special record of all nines, which prevents finding the true end of the file. The program converts the time passed into the number of seconds for the day. This field is used as part of the key to the file to simplify processing. If a request is made of all nines, the program ensures that a matching record exists in the file. If it is not in the file, it is added.

SBMTIMJOB Command and Processing Program

The following is the source for the SBMTIMJOB command:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00 /*PARMS PGM(SBMTIMJOBC)                                              */
  2.00 /*                                                                   */
  3.00 /*  Submit time job submits a command to be run at a later           */
  4.00 /*    time. The time can be now, a time later today or on a          */
  5.00 /*    specific date.                                                 */
  6.00 /*                                                                   */
  7.00 /*  The CPP is SBMTIMJOBC                                            */
  8.00 /*                                                                   */
  9.00 /*  A related command is SNDTIMMSG                                   */
 10.00 /*                                                                   */
 11.00          CMD         PROMPT('Submit Time Command')
 12.00          PARM        RQSDTA TYPE(*CHAR) LEN(256) MIN(1) EXPR(*YES) +
 13.00                         PROMPT('Request data:')
 14.00          PARM        HOUR TYPE(*CHAR) LEN(2) MIN(1) +
 15.00                         RANGE(00 24) SPCVAL(('*NOW' '**')) +
 16.00                         PROMPT('Hour as 24 hr clock or *NOW:')
 17.00          PARM        MIN TYPE(*CHAR) LEN(2) DFT(00) +
 18.00                         RANGE(00 59) +
 19.00                         PROMPT('Minute (00-59):')
 20.00          PARM        DATE TYPE(*DEC) LEN(6 0) RANGE(000000 999999) +
 21.00                         DFT(*TODAY) SPCVAL((*TODAY 0)(*DAY1 -1) +
 22.00                         (*DAY2 -2)(*DAY3 -3)(*DAY4 -4)(*DAY5 -5) +
 23.00                         (*DAY6 -6)(*DAY7 -7)(*DAY8 -8)(*DAY9 -9) +
 24.00                         (*DAY10 -10)(*DAY11 -11)(*DAY12 -12) +
 25.00                         (*DAY13 -13)(*DAY14 -14)) +
 26.00                         PROMPT('Date (sys fmt) *TODAY *DAYnn:')
 27.00          PARM        JOB TYPE(*NAME) LEN(10) DFT(*JOBD) +
 28.00                         SPCVAL(*JOBD) EXPR(*YES) +
 29.00                         PROMPT('Job name:')
 30.00          PARM        JOBD TYPE(QUAL1) +
 31.00                         PROMPT('Job description:')
 32.00          PARM        DSPSBMJOB TYPE(*CHAR) LEN(4) DFT(*YES) +
 33.00                         RSTD(*YES) VALUES(*YES *NO) +
 34.00                         PROMPT('Allow display by DSPSBMJOB?:')
 35.00 QUAL1:   QUAL        TYPE(*NAME) LEN(10) EXPR(*YES) DFT(QBATCH)
 36.00          QUAL        TYPE(*NAME) LEN(10) DFT(*LIBL) SPCVAL(*LIBL) +
 37.00                         EXPR(*YES) PROMPT('Library:')
```

The following is the source for the SBMTIMJOBC CL program:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00 /* SBMTIMJOBC - Submit time job - CPP for SBMTIMJOB                */
  2.00            PGM        PARM(&RQSDTA &HR &MIN &DATEI &JOB &JOBDF +
  3.00                         &WRKSBMJOB)
  4.00            DCL        &RQSDTA *CHAR LEN(256)
  5.00            DCL        &RLSCMD *CHAR LEN(256)
  6.00            DCL        &SBM100 *CHAR LEN(100)
  7.00            DCL        &HR *CHAR LEN(2)
  8.00            DCL        &MIN *CHAR LEN(2)
  9.00            DCL        &DATEI *DEC LEN(6 0) /* Input date */
 10.00            DCL        &JOB *CHAR LEN(10)
 11.00            DCL        &JOBDF *CHAR LEN(20) /* Full Job description */
 12.00            DCL        &WRKSBMJOB *CHAR LEN(4)
 13.00            DCL        &JOBD *CHAR LEN(10)
 14.00            DCL        &JOBDLIB *CHAR LEN(10)
 15.00            DCL        &TIME *CHAR LEN(6)
 16.00            DCL        &DATEO *CHAR LEN(6) /* Output date */
 17.00            DCL        &DATEJA *CHAR LEN(5) /* Julian alpha date */
 18.00            DCL        &DATEJN *DEC LEN( 5 0) /* Julian dec date */
 19.00            DCL        &CURDAT *CHAR LEN(6) /* Current date */
 20.00            DCL        &CURDATJA *CHAR LEN(6) /* Julian current date */
 21.00            DCL        &DAYS *DEC LEN(3 0)
 22.00            DCL        &DAYSA *CHAR LEN(3)
 23.00            DCL        &YEAR *DEC LEN(2 0)
 24.00            DCL        &YEARA *CHAR LEN(2)
 25.00            DCL        &LEAP *CHAR LEN(18) VALUE('000408121684889296')
 26.00            DCL        &FLDLEN *DEC LEN(5 0) VALUE(2)
 27.00            DCL        &X *DEC LEN(3 0) VALUE(-1) /* Index to leap */
 28.00            DCL        &ENDYR *DEC LEN(3 0) VALUE(365)
 29.00            DCL        &DATEA *CHAR LEN(6)  /* Date in alpha */
 30.00            DCL        &CURHR *CHAR LEN(2)
 31.00            DCL        &CURMIN *CHAR LEN(2)
 32.00            DCL        &WRKSTN *CHAR LEN(10) /* Requester work stn */
 33.00            DCL        &USER *CHAR LEN(10)
 34.00            DCL        &JOBNBR *CHAR LEN(6)
 35.00            DCL        &NOW *LGL
 36.00            DCL        &MSGID *CHAR LEN(7)
 37.00            DCL        &MSGDTA *CHAR LEN(132)
 38.00            RTVJOBA    JOB(&WRKSTN) USER(&USER) NBR(&JOBNBR)
 39.00            RTVSYSVAL  QHOUR RTNVAR(&CURHR)
 40.00            RTVSYSVAL  QMINUTE RTNVAR(&CURMIN)
 41.00            RTVSYSVAL  QDATE RTNVAR(&CURDAT)
 42.00            IF         (&HR *EQ '**') DO /* *NOW converted to ** */
 43.00            CHGVAR     &HR &CURHR
 44.00            CHGVAR     &MIN &CURMIN
 45.00            CHGVAR     &NOW '1' /* Set to 'on' */
 46.00            ENDDO      /* End *NOW converted to ** */
 47.00            CHGVAR     &DATEA &CURDAT  /* Start with today's date */
 48.00            IF         (&DATEI *GT 0) CHGVAR &DATEA &DATEI /* Value */
 49.00            IF         (&DATEI *LT 0) DO /* *DAYnn function */
 50.00            CVTDAT     &DATEA TOVAR(&DATEJA) TOFMT(*JUL) TOSEP(*NONE)
 51.00            CHGVAR     &DATEJN &DATEJA  /* Change to numeric */
 52.00            CHGVAR     &DATEJN (&DATEJN - &DATEI) /* Sub neg nbr */
 53.00            CHGVAR     &DATEJA &DATEJN  /* Change to alpha */
```

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
 54.00                     /* Check for date which is in the next year */
 55.00           CHGVAR    &YEARA %SST(&DATEJA 1 2) /* Extract year */
 56.00           CHGVAR    &DAYSA %SST(&DATEJA 3 3) /* Extract days */
 57.00           CHGVAR    &YEAR &YEARA /* Change to numeric */
 58.00           CHGVAR    &DAYS &DAYSA /* Change to numeric */
 59.00 LEAPLOOP: CHGVAR    &X (&X + 2) /* Index to leap year years */
 60.00           IF        (&X *LT 19) DO /* &X is less than 19 */
 61.00           IF        (%SST(&LEAP &X 2) *EQ &YEARA) CHGVAR &ENDYR 366
 62.00           GOTO      LEAPLOOP
 63.00           ENDDO     /* &X is less than 19 */
 64.00           IF        (&ENDYR *LT &DAYS) DO /* Bump for next year */
 65.00           CHGVAR    &DAYS (&DAYS - &ENDYR) /* Subtract end year */
 66.00           CHGVAR    &YEAR (&YEAR + 1) /* Increment year */
 67.00           CHGVAR    &YEARA &YEAR /* Change to alpha */
 68.00           CHGVAR    &DAYSA &DAYS /* Change to alpha */
 69.00           CHGVAR    %SST(&DATEJA 1 2) &YEARA /* Move new days */
 70.00           CHGVAR    %SST(&DATEJA 3 3) &DAYSA /* Move new year */
 71.00           ENDDO     /* Bump for next year */
 72.00           CVTDAT    &DATEJA FROMFMT(*JUL) TOVAR(&DATEA) TOSEP(*NONE)
 73.00           ENDDO     /* End of *DAYnn function */
 74.00           CVTDAT    &DATEA TOVAR(&DATEO) TOFMT(*YMD) TOSEP(*NONE)
 75.00           MONMSG    MSGID(CPF0000) EXEC(GOTO BADDATE)
 76.00           CVTDAT    &CURDAT TOVAR(&CURDATJA) TOFMT(*YMD) TOSEP(*NONE)
 77.00           IF        (&CURDATJA *GT &DATEO) GOTO BADDATE /* Prior */
 78.00           IF        (&CURDATJA *EQ &DATEO) DO /* Current date */
 79.00           IF        ((&CURHR *GT &HR) *OR ((&CURHR *EQ &HR) *AND +
 80.00                         (&CURMIN *GT &MIN))) GOTO BADTIME
 81.00           ENDDO     /* End of current date */
 82.00 QCMDCHK:  CALL      QCMDCHK PARM(&RQSDTA 230) /* Check syntax */
 83.00           MONMSG    MSGID(CPF6801) EXEC(GOTO CMDCNL) /* F3 used */
 84.00           MONMSG    MSGID(CPF0000) EXEC(GOTO BADCMD)
 85.00           CHGVAR    &SBM100 &RQSDTA
 86.00           CHGVAR    &JOBD %SST(&JOBDF 1 10)
 87.00           CHGVAR    &JOBDLIB %SST(&JOBDF 11 10)
 88.00           IF        (&JOBD *EQ '*USRPRF') DO /* *USRPRF */
 89.00           SBMJOB    JOB(&JOB) JOBD(*USRPRF) +
 90.00                         RQSDTA(&RQSDTA) HOLD(*YES) +
 91.00                         DSPSBMJOB(&WRKSBMJOB)
 92.00           MONMSG    CPF0000 EXEC(GOTO BADSBM)
 93.00           ENDDO     /* *USRPRF */
 94.00           IF        (&JOBD *NE '*USRPRF') DO /* Not *USRPRF */
 95.00           SBMJOB    JOB(&JOB) JOBD(&JOBDLIB/&JOBD) +
 96.00                         RQSDTA(&RQSDTA) HOLD(*YES) +
 97.00                         DSPSBMJOB(&WRKSBMJOB)
 98.00           ENDDO     /* Not *USRPRF */
 99.00 RCVMSG:   RCVMSG    MSGID(&MSGID) MSGDTA(&MSGDTA) MSGTYPE(*COMP)
100.00           IF        ((&MSGID *NE CPC1221) *AND (&MSGID *NE ' ')) +
101.00                          GOTO RCVMSG /* Not CPC 1221 */
102.00           IF        (&MSGID *EQ CPC1221) THEN(DO) /* SBMJOB cmpl */
103.00           SNDPGMMSG MSGID(CPC1221) MSGF(QCPFMSG) MSGTYPE(*COMP) +
104.00                         MSGDTA(&MSGDTA) /* Send completion msg */
105.00           CHGVAR    &TIME (&HR *CAT &MIN *CAT '00') /* Build time */
106.00           CHGVAR    &RLSCMD ('RLSJOB JOB(' *CAT %SST(&MSGDTA 21 6) +
107.00                         *TCAT '/' *TCAT %SST(&MSGDTA 11 10) +
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
                                 *TCAT '/' *TCAT %SST(&MSGDTA 1 10) +
                                 *TCAT ')')  /* Build RLSJOB command */
                  CALL       TIMADD PARM(&DATEO &TIME &RLSCMD &WRKSTN +
                                 &USER &JOBNBR &SBM100) /* Add to TIMFILP */
                  CALL       QSNDDTAQ PARM(TIMDTAQ QGPL &FLDLEN 'AA')
                  IF         &NOW SNDPGMMSG MSG('Job will not be held')
                  ELSE       SNDPGMMSG MSG('Job will be held until +
                                 time delay')
                  ENDDO      /* End of CPC1221 completion msg */
                  ELSE       SNDPGMMSG MSGID(CPF9898) MSGF(QCPFMSG) +
                                 MSGTYPE(*DIAG) MSGDTA('CPC1221 message +
                                 not received from SBMJOB command - job +
                                 will not be released')
                  RETURN
       BADDATE:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                                 MSGDTA('Date is prior to current system +
                                 date or it is invalid')
       BADTIME:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                                 MSGDTA('Hour/Minute is less than or +
                                 equal to the current system time')
       BADCMD:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                                 MSGDTA('Syntax of the command entered +
                                 is invalid.  See low level messages')
       BADSBM:   SNDPGMMSG  MSGID(CPF9898) MSGF(QCPFMSG) MSGTYPE(*ESCAPE) +
                                 MSGDTA('SBMJOB command failed.  See +
                                 See low level messages')
       CMDCNL:   SNDPGMMSG  MSGID(CPF6801) MSGF(QCPFMSG) MSGTYPE(*ESCAPE)
                 ENDPGM
```

The SBMTIMJOBC program accepts the parameters passed from the command and determines the date to be used. The &DATEI variable is handled identically to that described in the SNDTIMMSGC program.

The program calls the system-provided program QCMDCHK to check the syntax of the command. If the user had entered the command as ?XXXXX, the QCMDCHK program causes the command to be prompted. Two conditions are monitored. CPF6801 is signaled if the F3 key is pressed during prompting. If any error occurs, the generic CPF0000 signals a syntax error.

The program uses the SBMJOB command to submit the command. It sends any escape messages sent by the SBMJOB command to the user. If none occurs, it receives the low level messages and looks for CPF1221. This is the SBMJOB completion message, which is sent to the requester.

The program builds the RLSJOB command to be run and passes it and control parameters to the TIMADD program, which adds the request to the TIMFILP file. The program sends an entry to the TIMDTAQ data queue to activate the request-processing program. This is done by a call to the QSNDDTAQ OS/400-supplied program.

The error handling on the return from QCMDCHK is probably sufficient if most of the commands are entered using the ?XXXXX function to call prompting. You can provide better message responses in the event of errors by retrieving the low level messages. The *CL Programmer's Guide* contains more information on messages.

You may want to change this command to allow for a specific time or a delay time value. You might want to include a special value in the HOUR parameter, to have the command run at 2:00 a.m. the following day (or whenever the system is not likely to be used). You may also want to add to the SBMTIMJOB command some of the other parameters that are supported on the SBMJOB command. You could also provide your own defaults for certain of these parameters and make them identical to the current job (for example, LIBL, OUTQ).

Request-Processing Program TIMPGMC/TIMPGM

The following is the source for the TIMPGMC CL program:

```
*... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
/* TIMPGMC - Time dependent scheduling - Sbmtd by SBMJOB command */
             PGM
             DCL        &FIELD *CHAR LEN(2)
             DCL        &FLDLEN *DEC LEN(5 0) VALUE(2)
             DCL        &DATEW *CHAR LEN(6) /* Work field */
             DCL        &CURDAT *CHAR LEN(6) /* Current date */
             DCL        &DATNOW *CHAR LEN(6) /* Used by 1st job */
             DCL        &DATPRV *CHAR LEN(6) /* Used by 1st job */
             DCL        &ACTION *CHAR LEN(1) VALUE('1')
             DCL        &TIME *CHAR LEN(6)
             DCL        &YYMMDD *CHAR LEN(6)
             DCL        &JOBNBR *CHAR LEN(6)
             DCL        &CMD *CHAR LEN(256)
             DCL        &WAIT *DEC LEN(5 0)  /* Wait time */
             DCL        &WAITA *CHAR LEN(5)  /* Wait time */
             DCL        &RTNCDE *CHAR LEN(1) /* Return from RPG */
             DCL        &CMDDTA *CHAR LEN(128)
             CHGJOB     LOG(4 0 *NOLIST)
                        /* Remove deleted records and initialize */
             CHKOBJ     OBJ(TIMFILP2.QGPL) OBJTYPE(*FILE)
             MONMSG     MSGID(CPF9800) EXEC(DO) /* Not found */
             CPYF       TIMFILP TOFILE(TIMFILP2.QGPL) CRTFILE(*YES) +
                          NBRRCDS(1) /* Create file for copying */
             ENDDO      /* Not found */
             CPYF       FROMFILE(TIMFILP) TOFILE(TIMFILP2.QGPL) +
                          MBROPT(*REPLACE) FROMRCD(1) COMPRESS(*NO)
             MONMSG     MSGID(CPF2817) CMPDTA(CPF2869) /* Ignore 0 */
             CPYF       FROMFILE(TIMFILP2.QGPL) TOFILE(TIMFILP) +
                          MBROPT(*REPLACE) INCREL((*IF DLTCOD *EQ ' '))
             MONMSG     MSGID(CPF2817) CMPDTA(CPF2869) /* Ignore 0 */
             CLRPFM     TIMFILP2.QGPL
             CALL       TIMADD PARM('999999' '      ' &CMD +
                          ' '  ' '  ' ' ' ')
             DLTDTAQ    TIMDTAQ.QGPL
             MONMSG     MSGID(CPF2105) /* Does not exist */
             CRTDTAQ    TIMDTAQ.QGPL MAXLEN(2) TEXT('Time dependent +
                          scheduling data queue')
             GRTOBJAUT  TIMDTAQ.QGPL OBJTYPE(*DTAQ) USER(*PUBLIC) +
                          AUT(*OBJEXIST)
/* The following is special code to add a record to the JDAYP     */
/*   if it is before 8:00 AM. This is in case the TIMPGMC pgm     */
/*   is not running at midnight. The ADDJAYD Pgm does not add     */
/*   a record to JDAYP if one already exists for the same date.   */
/*                                                                */
             RTVSYSVAL  SYSVAL(QTIME) RTNVAR(&TIME)
             IF         (&TIME *LT '080000') DO /* Before 8:00 AM */
             RTVJOBA    NBR(&JOBNBR)
             RTVSYSVAL  QDATE RTNVAR(&CURDAT)
             CVTDAT     DATE(&CURDAT) TOVAR(&YYMMDD) TOFMT(*YMD) +
                          TOSEP(*NONE)
             CALL       ADDJDAY PARM(&YYMMDD &JOBNBR)
             ENDDO      /* Before 8:00 AM */
/*                                                              */
```

```
*... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
            GOTO       TIMPGM /* Check for first due request */
                       /* Wait for a message in TIMDTAQ */
 WAIT:      CHGVAR     &WAITA &WAIT
            SNDPGMMSG  MSG(&WAITA *CAT ' wait time in seconds')
            CALL       QRCVDTAQ PARM(TIMDTAQ QGPL &FLDLEN &FIELD +
                         &WAIT)
 /*  The following code is to submit a special job at midnight      */
            RTVSYSVAL  QDATE RTNVAR(&DATNOW)
            IF         (&DATNOW *NE &DATPRV) DO /* 1st job of day */
            CHGVAR     &DATPRV &DATNOW
                       /* Must run under profile with all obj auth */
            SBMJOB    JOB(MIDNIGHT) JOBD(MIDNIGHT.TESTARC) +
                        USER(*JOBD) RQSDTA('CALL MIDNIGHTC') +
                        DSPSBMJOB(*NO)
            ENDDO      /* 1st job of day */
 /*  End of special code for midnight job                           */
            SNDPGMMSG  MSG('Message ' *CAT &FIELD *CAT ' arrived on +
                         on TIMDTAQ queue')
            IF         (&FIELD *EQ 'LR') CHGVAR &ACTION '2'
 TIMPGM:    RTVSYSVAL  QDATE RTNVAR(&DATEW)
            CVTDAT     &DATEW TOVAR(&CURDAT) TOFMT(*YMD) TOSEP(*NONE)
            CALL       TIMPGM PARM(&ACTION &RTNCDE &WAIT &CURDAT +
                         &CMD)
            IF         (&RTNCDE *EQ '1')  DO /* Run cmd */
                       /* Write cmd to job log as an audit trail */
            CHGVAR     &CMDDTA %SST(&CMD 1 128) /* 1st half of cmd */
            SNDPGMMSG  MSG(&CMDDTA)
            CHGVAR     &CMDDTA %SST(&CMD 129 128) /* 2nd half */
            IF         (&CMDDTA *NE '     ') SNDPGMMSG MSG(&CMDDTA)
            CALL       QCAEXEC PARM(&CMD 256)
            MONMSG     MSGID(CPF1321) EXEC(SNDMSG MSG('Time +
                         Scheduling received CPF1321 (No such job) +
                         for ' *CAT %SST(&CMD 1 45)) TOMSGQ(QSYSOPR))
            MONMSG     MSGID(CPF1349) EXEC(SNDMSG MSG('Time +
                         Scheduling received CPF1349 (Not held) +
                         for ' *CAT %SST(&CMD 1 45)) TOMSGQ(QSYSOPR))
            GOTO       TIMPGM  /* Check for more to be run */
            ENDDO      /* Run cmd */
            IF         (&RTNCDE *EQ '2')  GOTO WAIT
                       /* RTNCDE = 3 is end of pgm - Fall out */
            ENDPGM
```

The TIMPGMC program begins by copying the TIMFILP file into QTEMP. It then copies TIMFILP back from QTEMP, but selects only those records with a blank in the DLTCOD field. It ensures that the nines record exists in the TIMFILP file by calling the TIMADD program with the special value of all nines. The program ensures the TIMDTAQ data queue exists in QGPL. The DLTDTAQ command is used to ensure the data queue does not grow beyond the minimal amount of space required. See the *CL Programmer's Guide*. The program retrieves the current date and calls TIMPGM to determine what to do. If the return code is 1, the command in the returned parameter should be run. If the return code is 2, no request is due. The TIMPGM program calculates the amount of time to wait before the next request is due and it uses that value in the parameter list for the QRCVDTAQ program. If the return code is 3, the program has received the special LR message and should end.

Assume that the SBMTIMJOB has already submitted a held job. When the job is due, the TIMPGMC program will attempt to release it by using the RLSJOB command as the parameter to CALL QCMDEXC. The MONMSG command for CPF1321 monitors for the situation in which the job no longer exists. The MONMSG command for CPF1349 monitors for the situation in which the job is no longer held. This means the job is still on the system but is either active or on the output queue. These situations could occur when the job is released independently of the TIMPGMC program or ended by the operator. In either case, a message is sent to QSYSOPR.

Other error conditions would cause the program to end abnormally on the CALL to QCMDEXC. For example, the SNDTIMMSG command ensures that the message queue exists when the command is entered, but it may not exist when the message is sent.

If the program abnormally ends, a program dump could be provided by using the job description INQMSGRPY parameter. You may wish to add some additional error handling. For example, you might choose also to monitor for CPF0000, and if it occurs, issue a DMPCLPGM command and send a message to QSYSOPR requesting that a programmer investigate. This will keep the program active and still allow problem analysis.

The following is the source for the TIMPGM RPG program:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
  1.00      F* TIMPGM - Acts on time functions - Called by TIMPGMC
  2.00      FTIMFILP UF  E           K        DISK
  3.00      ISAVTIM      DS
  4.00      I                                        1   20HR
  5.00      I                                        3   40MIN
  6.00      I                                        5   60SEC
  7.00      C           *ENTRY    PLIST
  8.00      C                     PARM           ACTION  1
  9.00      C                     PARM           RTNCDE  1
 10.00      C                     PARM           WAIT    50
 11.00      C                     PARM           CURDAT  6
 12.00      C                     PARM           CMD   256
 13.00      C* Action 2 requests end of program
 14.00      C           ACTION    IFEQ '2'                        Action 2
 15.00      C                     MOVE '3'       RTNCDE           End Pgm
 16.00      C                     SETON                     LR    Set LR
 17.00      C                     RETRN                           Return
 18.00      C                     END                             Action 2
 19.00      C* Action 1 checks file to see if a record is due
 20.00      C           ' '       SETLLTIMFILR                    Reset
 21.00      C           RDAGAN    TAG                             RDAGAN
 22.00      C                     READ TIMFILR                  20 EOF
 23.00      C* EOF should never occur
 24.00      C           CURDAT    IFLT RQSDAT                     Future day
 25.00      C                     EXSR CURSEC                     Calc TOTSEC
 26.00      C                     MOVE '2'       RTNCDE           Wait
 27.00      C           86401     SUB  TOTSEC    WAIT             End of day
 28.00      C                     EXCPTRELEAS                     Release lock
 29.00      C                     RETRN                           Return
 30.00      C                     END                             Future day
 31.00      C           CURDAT    IFEQ RQSDAT                     Today
 32.00      C                     EXSR CURSEC                     Calc TOTSEC
 33.00      C           TOTSEC    IFLT NBRSEC                     Not yet
 34.00      C           NBRSEC    SUB  TOTSEC    WAIT             Wait secs
 35.00      C                     MOVE '2'       RTNCDE           Wait
 36.00      C                     EXCPTRELEAS                     Release lock
 37.00      C                     RETRN                           Return
 38.00      C                     END                             Not yet
 39.00      C                     END                             Today
 40.00      C* Either past due date or today and past time - run
 41.00      C                     MOVE '1'       RTNCDE           Run
 42.00      C                     MOVE 'X'       DLTCOD           Delete cde
 43.00      C                     MOVE UDATE     SNTDAT           Sent date
 44.00      C                     MOVE CURTIM    SNTTIM           Sent time
 45.00      C                     UPDATTIMFILR                    Update rcd
 46.00      C                     RETRN                           Return
 47.00      C* Calculate TOTSEC which is seconds used so far today
 48.00      C           CURSEC    BEGSR
 49.00      C                     TIME           CURTIM  60       Get time
 50.00      C                     MOVE CURTIM    SAVTIM  6        Move to DS
```

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7
 51.00      C           3600      MULT HR        TOTSEC  50       Hrs to secs
 52.00      C           60        MULT MIN       SAV5N   50       Min to secs
 53.00      C                     ADD  SAV5N     TOTSEC           Hrs and mins
 54.00      C                     ADD  SEC       TOTSEC           Total secs
 55.00      C                     ENDSR
 56.00      OTIMFILR E                RELEAS
```

The TIMPGM program tests the code for the action to be performed. If the action code is 2, the program ends. If the action code is 1, the program accesses the first record in the file with a blank deletion code. The SETLL operation is used because the file should always be positioned to read the record with the lowest key in the file. Any record that has been added with a lower key (an earlier time) is now the first record and will be read first.

If no request is due that day, the program returns a value to wait until the end of the day. If a request is due that day but is not yet overdue, the program returns the difference in terms of number of seconds between the current time and the first record. This is used as the WAIT value on the QRCVDTAQ.

If a request is past the due date or is due today and past the current time, the record is updated as processed and the return code is set to cause the command to be run.

Ending Time-Dependent Scheduling

The time-dependent scheduling function should be ended by use of the ENDTIMSCH command. This command sends a special entry to the TIMDTAQ data queue.

The following is the source for the ENDTIMSCH command:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ...
  1.00       CMD        PROMPT('End Time Scheduling')
  2.00                  /* CPP is ENDTIMSCHC */
```

The following is the source for the ENDTIMSCHC CL Program:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ...
  1.00              PGM
  2.00              DCL        &FLDLEN *DEC LEN(5 0) VALUE(2)
  3.00              CALL       QSNDDTAQ PARM(TIMDTAQ QGPL &FLDLEN 'LR')
  4.00              SNDPGMMSG  MSG('Ending record sent to TIMDTAQ') +
  5.00                           MSGTYPE(*COMP)
  6.00              ENDPGM
```

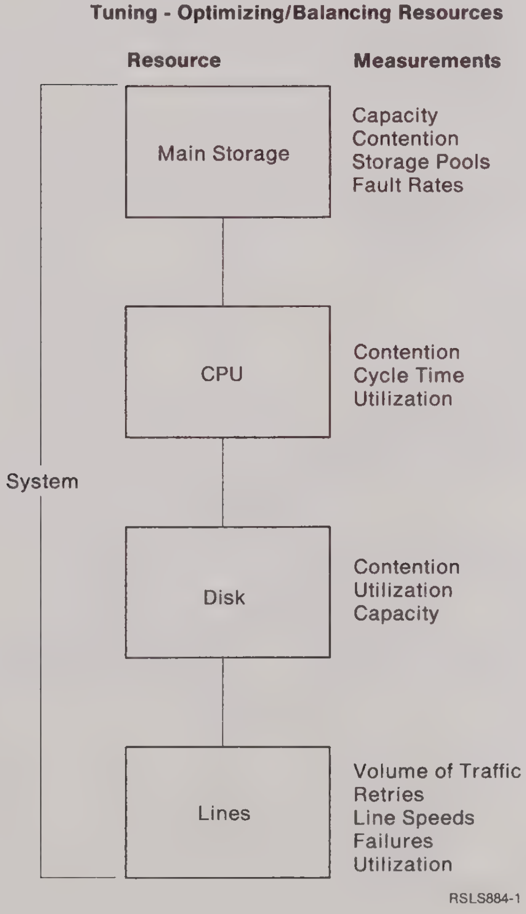
Additional Request Commands

The sample describes two requester commands for sending messages or submitting commands. You may want to add your own commands for specific functions to be run or submitted. You could consider using one of sample commands and processing programs as a base and making specific modifications.

Chapter 5. Performance Guidelines

The purpose of this chapter is to give you some general performance tuning information. This chapter will help you understand some of the work management parameters that affect performance and how to set up and analyze your system. Prior to using this chapter, you should set your performance objectives regarding interactive throughput, interactive response time, batch job run time, spool throughput, and so on.

Achieving good system performance requires a proper balance among all system resources as shown below:


Tuning - Optimizing/Balancing Resources
Resource
Measurements
System
Main Storage
Capacity
Contention
Storage Pools
Fault Rates
CPU
Contention
Cycle Time
Utilization
Disk
Contention
Utilization
Capacity
Lines
Volume of Traffic
Retries
Line Speeds
Failures
Utilization
RSLS884-1

Overutilizing any resource will affect performance. With the proper setting of the work management parameters, you can make the best use of each resource.

This chapter is organized into the following sections:

- Performance Overview, which describes:
  - Job states and transitions
  - Process access groups (PAG)

  - Purge attributes
  - Pools, activity levels, and ineligible queue management
  - Time slice

- Performance Commands, which describe the Work with System Status (WRKSYSSTS) and Work with Active Jobs (WRKACTJOB) commands

- Basic Tuning, which describes:

  - IPL performance adjustment
  - Estimating the machine pool size
  - Building an activity level for *BASE
  - When to choose more storage pools
  - How to estimate the initial size and activity levels

- Specialized Performance Tuning, which describes more complex tuning environments, for example:

  - Multiple pools for interactive or batch environments
  - Communications environments

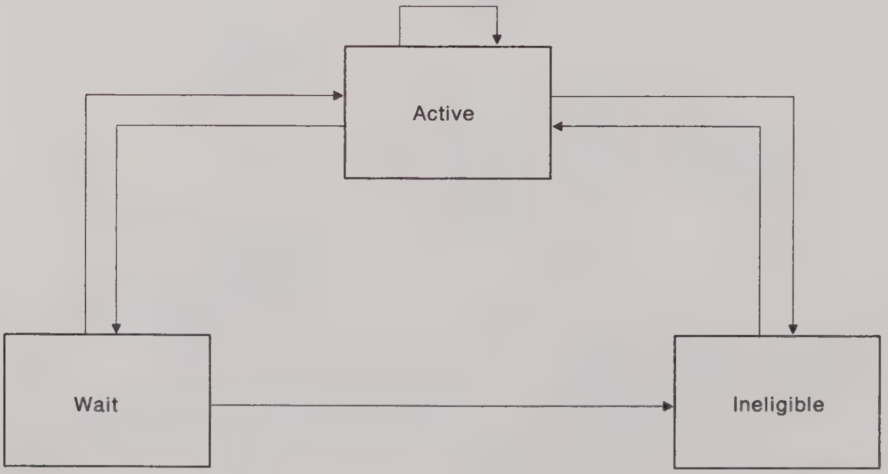
If performance tuning is new to you, you should probably just use the first 3 sections. If you are experienced on performance tuning and have a large environment (more than 100 active jobs), you will probably be most interested in "Specialized Tuning" on page 5-20.

Performance Overview

Job States

Jobs running in the system will be in any of the three states shown below:

Job States/Transitions



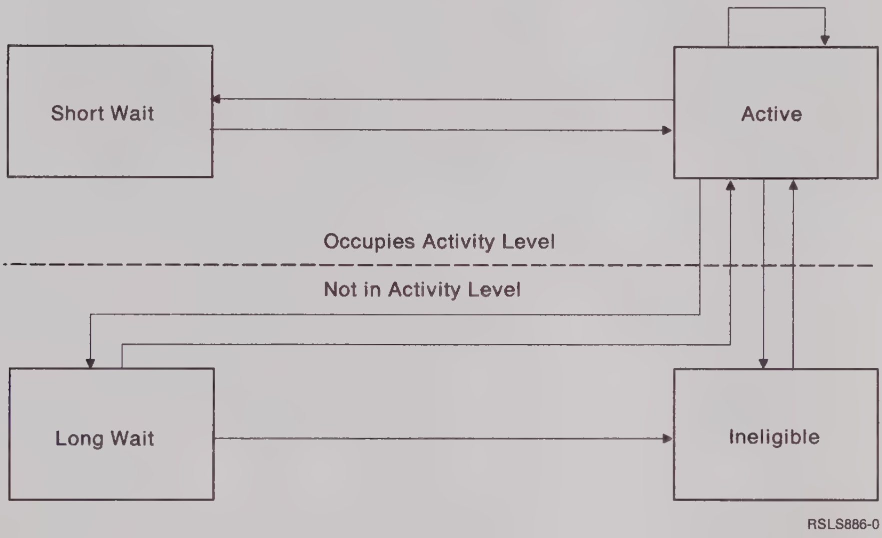
RSLS885-0

When a job is active, it is in main storage processing work requested by the application. If a job is waiting, it means that the job needs a resource that is not available. A job is ineligible when it has work to do, but the system is unable to accept more work at that time.

Wait States

When a job is waiting for a resource, it may wait in main storage or it may be removed from main storage until the object is available. There are two types of waits: short waits and long waits as shown below.

Job States/Transitions



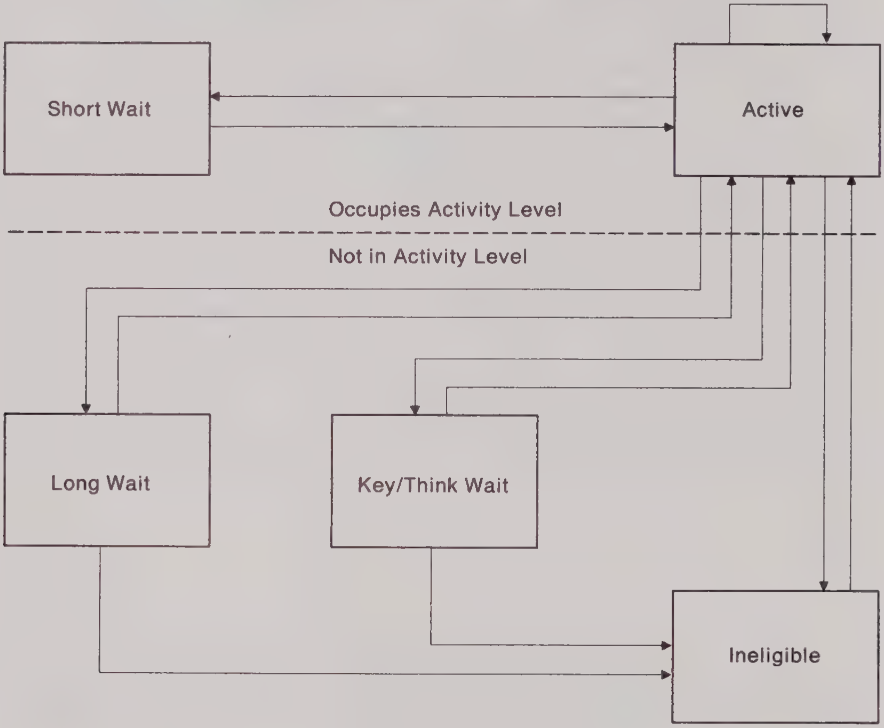
A short wait occurs in main storage and causes one of the available activity level positions to be unavailable until the requested resource is available or two seconds have ended. If the wait is longer than 2 seconds, it is changed to a long wait and the job's position may be taken by a job with something more productive to do. Some typical causes of short waits are:

- Sending a WRITE to a display with the DFRWRT(*NO) parameter
- Sending break messages to work stations
- Specifying RSTDSP(*YES) on display files

When using remote lines, these types of waits should be avoided because they cause the main storage resource to be used inefficiently.

Long waits take place outside of main storage and make the job activity level position available to other jobs. Long waits are usually caused by record lock conflicts. A specialized form of long wait is called key/think wait as shown below.

Job States/Transitions


Short Wait
Active
Occupies Activity Level
Not in Activity Level
Long Wait
Key/Think Wait
Ineligible

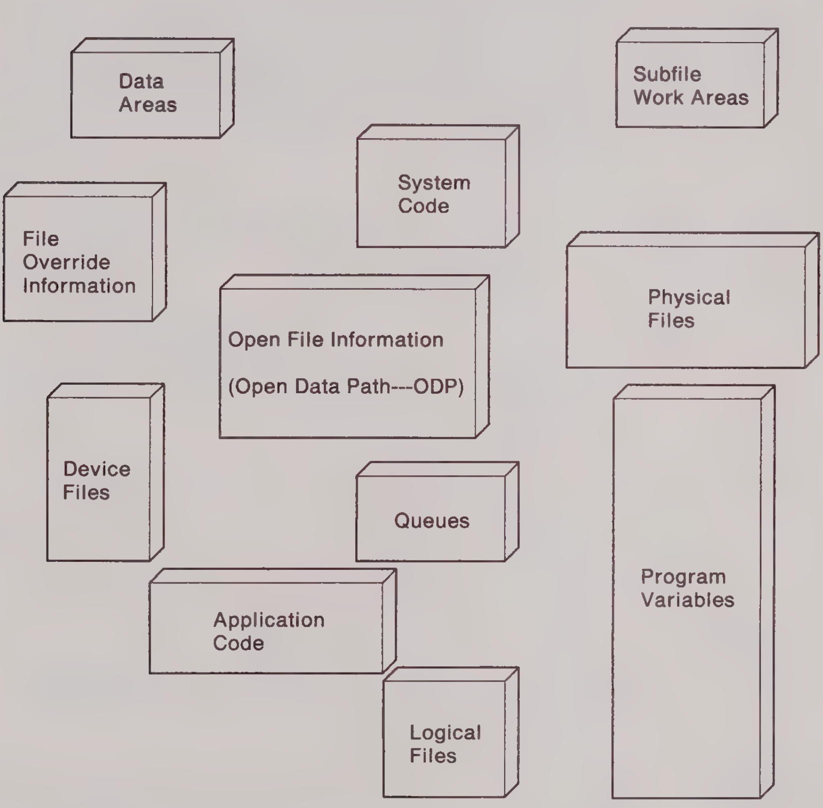
RSLS887-1

This wait also takes place outside the activity level and occurs when a job has completed a work assignment and returns to request more work. When the job receives a new assignment or gets tired of waiting (times out), it will then attempt to run again. If no activity level space is available, the job becomes ineligible.

System Objects/Process Access Groups (PAG)

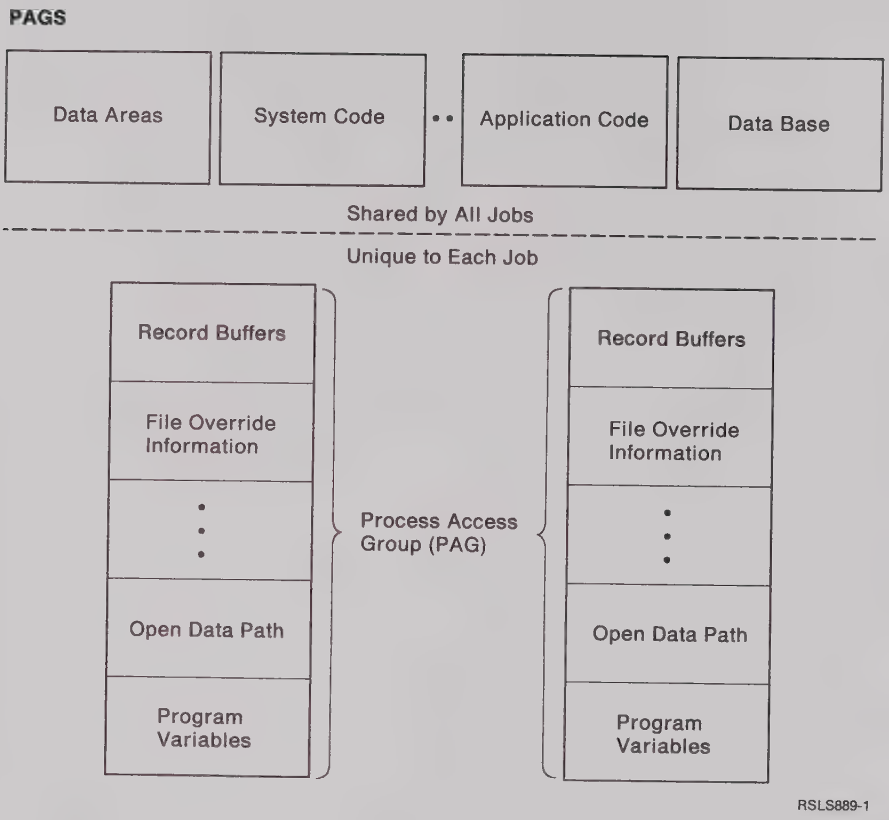
Each job running in the system is assigned a storage pool. When a job is active, it resides in its assigned storage pool. Jobs that are active are referring to many different system objects. Some of these objects used by jobs are:

Some Objects Used by Jobs


Data Areas
Subfile Work Areas
System Code
File Override Information
Physical Files
Open File Information
(Open Data Path---ODP)
Device Files
Queues
Program Variables
Application Code
Logical Files

RSLS888-1

As the job uses these objects, they must be in main storage. If they are not in main storage, they must be read into main storage from their locations on disk (auxiliary) storage. Although many different object types are used, they fall into two main categories. The object is either shared by jobs or is unique to a specific job. When an object is shared by jobs, only one copy of the object exists. For example, application code that is used by 20 jobs will reside in main storage in only one place but will be used by all jobs. However, the variables and data in the application will not have the same values for all jobs using the application. These portions of the application and other job unique objects are packaged as an object called a process access group (PAG).


PAGS
Data Areas
System Code
Application Code
Data Base
Shared by All Jobs
Unique to Each Job
Record Buffers
File Override Information
Open Data Path
Program Variables
Process Access Group (PAG)
Record Buffers
File Override Information
Open Data Path
Program Variables
RSLS889-1

Whenever a job is active, its PAG must be in main storage.

PAG Transfer

In order to get a job's PAG into main storage, the system refers to the PURGE parameter. The PURGE parameter is specified in the job class which is resolved when the job first enters the system. The value for the PURGE parameter is *YES or *NO. The characteristics of these values are:

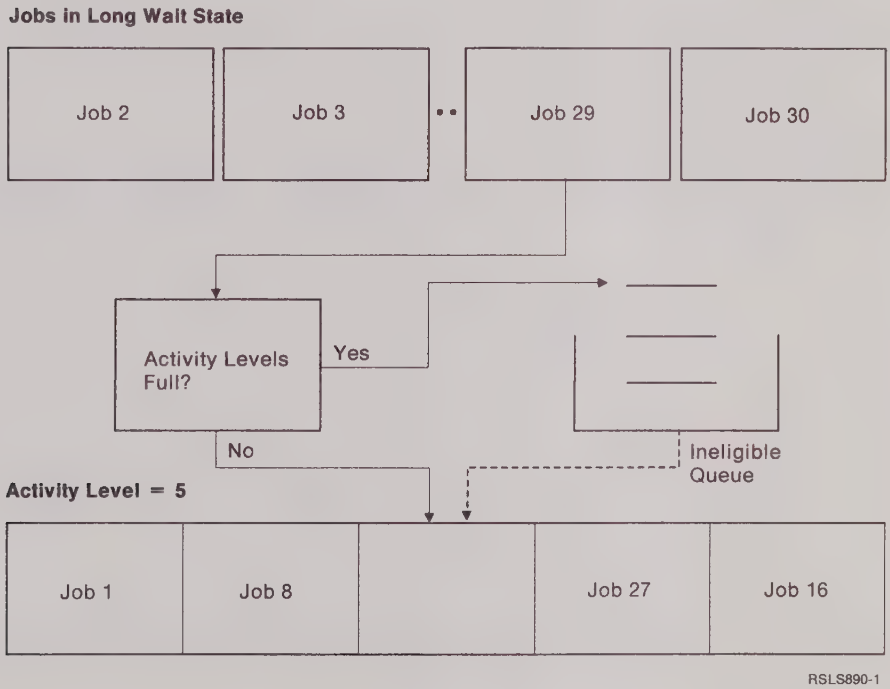
- *NO
  - Few nondata base reads per transaction
  - Few writes per transaction
  - Reads and writes transfer few pages
  - May reduce disk utilization
  - Less processing unit per transaction
  - Requires more main storage
- *YES
  - Adapts to workload/storage size
    - Operates as *YES in storage constrained or balanced workload environments
    - Operates as *NO in storage rich or imbalanced workload environments
  - Performs better with small main storage

In summary, using *YES relieves you from deciding which value to use; but, *NO may give you better performance.

Activity Levels and Ineligible Queue

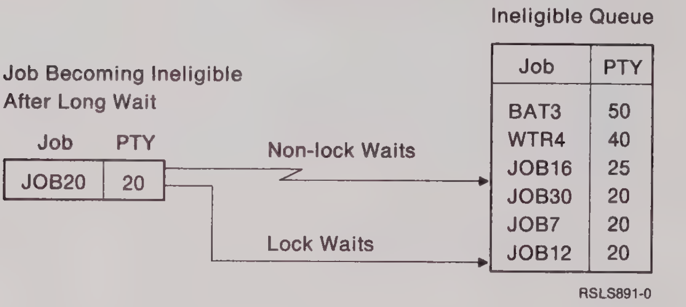
For interactive environments, there are typically more jobs running than there is space for them to run. When a job attempts to run, there must be space for the job in main storage. To restrict the number of jobs in main storage at the time, a value, called the activity level, is specified for each pool in the system. Before a job can become active, there must be an available activity level.

As shown below, when a job leaves a long wait condition, a test is made to determine if there is an available activity level. If an activity level is available, the job becomes active and begins processing in main storage. If no activity level is available, the job becomes ineligible. When a job becomes ineligible, it is placed in the ineligible queue until an activity level is available.


Jobs in Long Wait State
Job 2
Job 3
Job 29
Job 30
Activity Levels Full?
Yes
No
Ineligible Queue
Activity Level = 5
Job 1
Job 8
Job 27
Job 16
RSLS890-1

The following figure shows how jobs are placed on the ineligible queue. If the job entered the long wait condition by other than a lock conflict, it is placed behind all other jobs of equal priority already on the ineligible queue. This is called a first-in, first-out priority queue.

Ineligible Queue


Ineligible Queue
Job
PTY
BAT3
50
WTR4
40
JOB16
25
JOB30
20
JOB7
20
JOB12
20
Job Becoming Ineligible
After Long Wait
Job
PTY
JOB20
20
Non-lock Waits
Lock Waits
RSLS891-0

However, if a job becomes ineligible after a long wait caused by a lock conflict, it is placed in front of jobs of equal priority already on the ineligible queue. There are two main reasons for this change to normal queue placement:

- Since the job entered a long wait as a result of a lock conflict, we know it was active (in main storage) before the conflict occurred. If the wait was of short duration (and many are), you may be able to get the job back into an activity level before all of the objects the job was using are removed from main storage.
- When the job leaving the wait state has been granted the lock, the job now *owns* the object it had been waiting for. If other jobs on the ineligible queue want to use the same object, they must wait until the object is once again available. Therefore, you want jobs holding locks on objects to use them and make them available for other jobs to use. To accomplish this, the job moves ahead of any potential requesters.

By correctly managing the ineligible queue, the system may avoid unnecessary job transitions and disk operations. As a result, throughput and response time will remain more consistent.

Time Slice

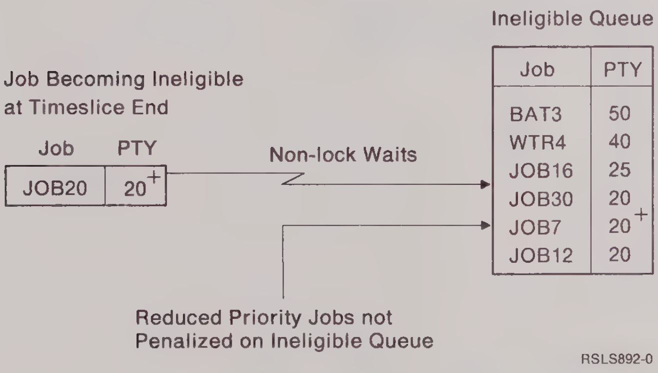
Another of the work management tuning parameters is time slice. The time slice value is specified in the job's class. This value represents the amount of processing unit time that a job may use while processing a transaction. It does not represent the elapsed time of a transaction.

If a job fails to complete a transaction in the specified time slice, the following will occur:

- The job's priority is reduced by 1/128.
- If no jobs of equal or higher priority are on the ineligible queue, the job is given another time slice, left in main storage, and allowed to continue running the transaction.
- If jobs of equal or higher priority are on the ineligible queue, the job running is removed from main storage and placed on the ineligible queue. A job from the ineligible queue is moved into main storage and processing continues.

Jobs placed on the ineligible queue as a result of exceeding their time slice value, are prioritized according to their original priority. The small reduction in priority will affect the job's ability to use the processing unit, but will not affect its ability to return to main storage, as shown below:

Ineligible Queue


Ineligible Queue
Job
PTY
BAT3 50
WTR4 40
JOB16 25
JOB30 20
JOB7 20+
JOB12 20
Job Becoming Ineligible at Timeslice End
Job
PTY
JOB20
20+
Non-lock Waits
Reduced Priority Jobs not Penalized on Ineligible Queue
RSLS892-0

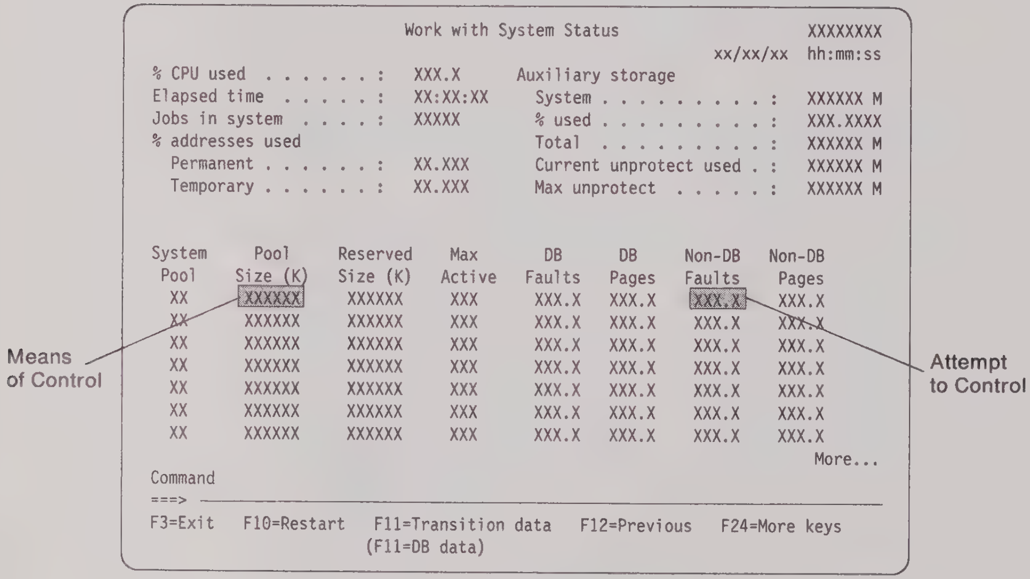
Regardless of the number of times the time slice is exceeded, the priority of the job is reduced only once. For interactive jobs, priority will be restored to its original value at the completion of each transaction. Since batch jobs do not have the concept of transactions, they will continue to run at a slightly lower priority until the job ends.

Performance Commands

When you are observing the performance of your system, there are two commands that are available to help you: Work with System Status (WRKSYSSTS) and Work with Active Jobs (WRKACTJOB). Also, the Performance Tools Licensed Program may be used to help analyze your performance. (The *Performance Tools Guide* contains this information.) However, this chapter discusses only the system commands.

Work with System Status (WRKSYSSTS) Command

The WRKSYSSTS command displays information about the current status of the system. When using the WRKSYSSTS display, you must separate the machine pool (Pool 1) from the other pools. Figure 5-1 and Figure 5-2 on page 5-11 show the WRKSYSSTS display.



```
                             Work with System Status                 XXXXXXXX
                                                          xx/xx/xx   hh:mm:ss
% CPU used . . . . . . :    XXX.X      Auxiliary storage
Elapsed time . . . . . :    XX:XX:XX     System . . . . . . . . . :   XXXXXX M
Jobs in system . . . . :    XXXXX        % used . . . . . . . . . :   XXX.XXXX
% addresses used                         Total  . . . . . . . . . :   XXXXXX M
  Permanent . . . . . . :   XX.XXX       Current unprotect used . :   XXXXXX M
  Temporary . . . . . . :   XX.XXX       Max unprotect  . . . . . :   XXXXXX M


System     Pool     Reserved    Max      DB       DB     Non-DB   Non-DB
 Pool    Size (K)   Size (K)   Active   Faults   Pages   Faults    Pages
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
  XX     XXXXXX     XXXXXX      XXX      XXX.X    XXX.X   XXX.X    XXX.X
                                                                    More...
Command
===> ____________________________________________________________________
F3=Exit   F10=Restart   F11=Transition data    F12=Previous    F24=More keys
                        (F11=DB data)
```

Means of Control

Attempt to Control

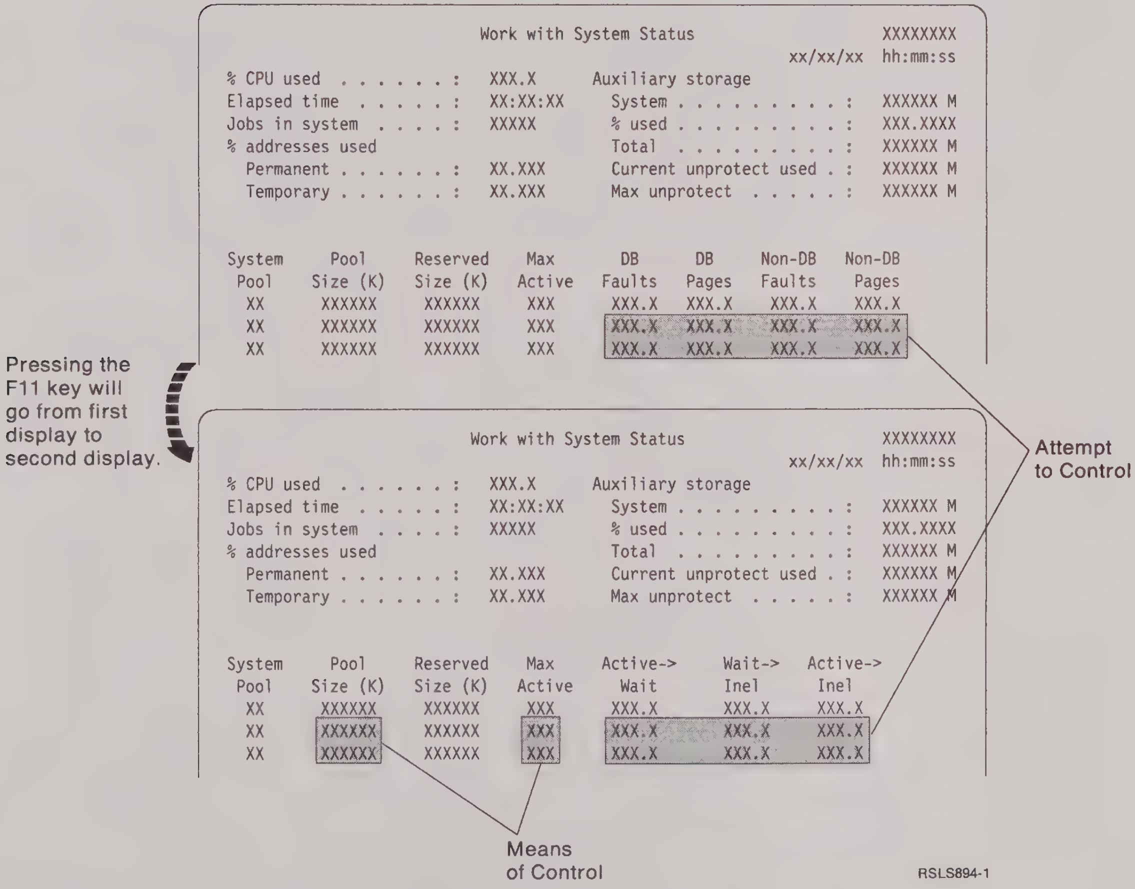
RSLS893-1

Figure 5-1. Example of the System Status (WRKSYSSTS) Display

Figure 5-1 on page 5-10 shows that our concern in the machine pool is the nondata base fault rate. Page faulting in the machine pool affects all jobs on the system. Therefore, it is desirable to maintain a low page fault rate in this pool. The guidelines that should be applied are:

	Nondata Base Paging Faults		
Main Storage Size	**Good**	**Acceptable**	**Poor**
Less than or equal to 12M	<2	2−5	>5
More than 12M	<1	1−3	>3

The only way to affect paging in the machine pool is to adjust the size of the pool.



```
                          Work with System Status                    XXXXXXXX
                                                          xx/xx/xx   hh:mm:ss
% CPU used  . . . . . . :    XXX.X      Auxiliary storage
Elapsed time  . . . . . :    XX:XX:XX     System . . . . . . . . . :   XXXXXX M
Jobs in system  . . . . :    XXXXX        % used . . . . . . . . . :   XXX.XXXX
% addresses used                          Total  . . . . . . . . . :   XXXXXX M
  Permanent . . . . . . :    XX.XXX       Current unprotect used . :   XXXXXX M
  Temporary . . . . . . :    XX.XXX       Max unprotect  . . . . . :   XXXXXX M


System     Pool      Reserved    Max       DB      DB      Non-DB    Non-DB
 Pool    Size (K)   Size (K)   Active    Faults   Pages    Faults    Pages
  XX      XXXXXX     XXXXXX     XXX      XXX.X    XXX.X    XXX.X     XXX.X
  XX      XXXXXX     XXXXXX     XXX      XXX.X    XXX.X    XXX.X     XXX.X
  XX      XXXXXX     XXXXXX     XXX      XXX.X    XXX.X    XXX.X     XXX.X
```

```
                         Work with System Status                     XXXXXXXX
                                                          xx/xx/xx   hh:mm:ss
% CPU used  . . . . . . :    XXX.X      Auxiliary storage
Elapsed time  . . . . . :    XX:XX:XX     System . . . . . . . . . :   XXXXXX M
Jobs in system  . . . . :    XXXXX        % used . . . . . . . . . :   XXX.XXXX
% addresses used                          Total  . . . . . . . . . :   XXXXXX M
  Permanent . . . . . . :    XX.XXX       Current unprotect used . :   XXXXXX M
  Temporary . . . . . . :    XX.XXX       Max unprotect  . . . . . :   XXXXXX M


System     Pool      Reserved    Max     Active->     Wait->    Active->
 Pool    Size (K)   Size (K)   Active     Wait        Inel       Inel
  XX      XXXXXX     XXXXXX     XXX      XXX.X       XXX.X      XXX.X
  XX      XXXXXX     XXXXXX     XXX      XXX.X       XXX.X      XXX.X
  XX      XXXXXX     XXXXXX     XXX      XXX.X       XXX.X      XXX.X
```

Pressing the F11 key will go from first display to second display.
Attempt to Control
Means of Control
RSLS894-1

Figure 5-2. Example of the System Status (WRKSYSSTS) Display

As Figure 5-2 on page 5-11 shows, all other pools require attention to both data base and nondata base page faulting rates and the job transitions. The guidelines for the faulting rates in these pools are based on the sum of the data base and nondata base faults per second. In addition, adjustments can be made for processor speed and the PURGE attribute. The general guidelines are:

	Sum of Data Base and Nondata Base Paging Faults		
Attribute/Model	**Good**	**Acceptable**	**Poor**
PURGE(*YES) Models A4, A6, A10, A20	<15	15−25	>25
PURGE(*YES) Models A30, A40	<20	20−30	>30
PURGE(*NO)	<25	25−30	>30

When observing job transitions:

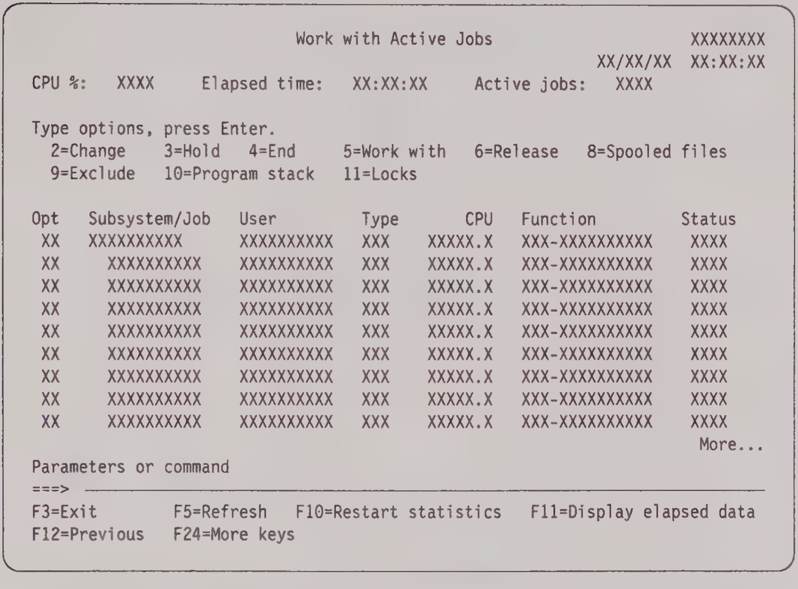
- Wait-to-ineligible transitions need not be 0 all the time. In heavy-use periods, it may be advisable to cause jobs to become ineligible in order to avoid excessive page fault rates.
- The active-to-ineligible transitions should be 0. If jobs are becoming ineligible, either the time slice is too small or the jobs are doing noninteractive type transactions.

Remember when using this display that page fault rates are much more important than the job transition values.

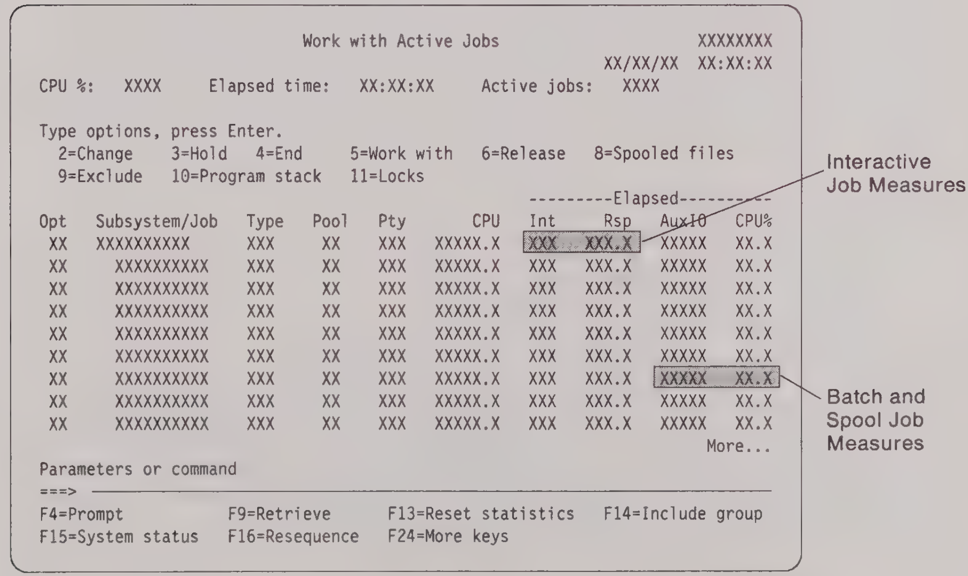
There are two methods to accomplish the desired values for pools 2-16. You can either increase or decrease the pool size or the activity level. The mechanics of these actions are discussed under "Basic Tuning" on page 5-14.

Work with Active Jobs (WRKACTJOB) Command

The WRKSYSSTS command is designed to give you an indication of system performance. The WRKACTJOB command helps you measure your system's performance. The *CL Reference* contains information on the format and description of the WRKACTJOB command. The following examples show the WRKACTJOB displays.



```
                              Work with Active Jobs                     XXXXXXXX
                                                              XX/XX/XX  XX:XX:XX
CPU %:   XXXX     Elapsed time:   XX:XX:XX     Active jobs:   XXXX

Type options, press Enter.
  2=Change    3=Hold   4=End     5=Work with   6=Release    8=Spooled files
  9=Exclude   10=Program stack   11=Locks

Opt   Subsystem/Job    User          Type        CPU   Function          Status
 XX   XXXXXXXXXX       XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
 XX     XXXXXXXXXX     XXXXXXXXXX    XXX     XXXXX.X   XXX-XXXXXXXXXX     XXXX
                                                                         More...
Parameters or command
===> ___________________________________________________________________________
F3=Exit        F5=Refresh    F10=Restart statistics    F11=Display elapsed data
F12=Previous   F24=More keys
```



```
                              Work with Active Jobs                     XXXXXXXX
                                                              XX/XX/XX  XX:XX:XX
CPU %:   XXXX     Elapsed time:   XX:XX:XX     Active jobs:   XXXX

Type options, press Enter.
  2=Change    3=Hold   4=End     5=Work with   6=Release    8=Spooled files
  9=Exclude   10=Program stack   11=Locks
                                                   ---------Elapsed----------
Opt   Subsystem/Job    Type   Pool   Pty       CPU   Int     Rsp   AuxIO   CPU%
 XX   XXXXXXXXXX       XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
 XX     XXXXXXXXXX     XXX     XX    XXX   XXXXX.X   XXX   XXX.X   XXXXX   XX.X
                                                                         More...
Parameters or command
===> ___________________________________________________________________________
F4=Prompt          F9=Retrieve      F13=Reset statistics    F14=Include group
F15=System status  F16=Resequence   F24=More keys
```

Interactive Job Measures

Batch and Spool Job Measures

RSLS896-0

When attempting to observe your system's performance, both WRKSYSSTS and WRKACTJOB should be used. With each observation period, you should examine and evaluate the measures of system performance against the goals you have set. Some of the typical measures include:

- Interactive throughput and response time. Available from the WRKACTJOB display.
- Batch throughput. Observe the `AuxIO` and `CPU%` values for active batch jobs.
- Spool throughput. Observe the `AuxIO` and `CPU%` values for active writers.

Each time you make tuning adjustments, you should measure and compare all of your key performance measures.

Basic Tuning

This section describes some of the steps taken to initially configure the system pool sizes and activity levels. The following topics are discussed and should help you tune your system efficiently:

- IPL performance adjustments
- Initial machine pool size
- Choosing your pool configuration
- Adjustments to pool sizes and activity levels

Initial Program Load (IPL) Performance Adjustments

If you would like the system to do initial tuning for you, leave the system value QPFRADJ at 1. When you IPL, the system examines the machine configuration and the controlling subsystem value. If QPFRADJ is set to 1, the system will use the configuration information to set the initial pool sizes and activity levels. If the controlling subsystem is QBASE or QCTL, separate pools are set up for spool and interactive jobs.

The IPL tuner sets the following:

- Machine pool size
- Base pool size and activity level
- Interactive pool size and activity level
- Spool pool size and activity level

Batch jobs (including S/36 environment evoked batch jobs) will run in the base pool. If you make any adjustments to the pool size or activity level values, you should consider setting QPFRADJ to 0. Otherwise, the values will be reset at the next IPL. If you are satisfied with the QPFRADJ values, but change your environment to run night batch jobs, you may wish to leave QPFRADJ set to 1.

Initial Machine Pool Size

As mentioned earlier, maintaining low page fault rates in this pool will help the system perform better. When setting the initial machine pool size, use the following:

```
S = M + J + L + F
```

where:

S = Initial machine pool size
M = Minimum machine pool size (Figure 5-3)
J = Job space (Figure 5-4)
L = Communications space (Figure 5-5)
F = Functional space (Figure 5-6)

To find the value for the minimum machine pool size, locate your main storage size in the table below and use the corresponding value. This is the same value the IPL tuner uses to calculate the machine pool size.

Main Storage Size (M)	Minimum Machine Pool Size (K)
4	1175
8	1625
12	2050
16	2400
20	2750
24	3050
28	3350
32	3650
36	3950
40	4250
48	4800
64	6200
80	7600
96	9000

Figure 5-3. Minimum Machine Pool Size

Job space is set aside in the machine pool for each active job in the system. In order to get a value for this part of the system, you can use the table below. However, a more accurate value will be derived if you:

- Estimate the maximum number of jobs (including group jobs) that will be active at the same time.
- Add the value you estimated to the value obtained from Figure 5-3 since each job requires 1K in the machine pool.

Main Storage Size (M)	Model A4	Model A6	Model A10	Model A20	Model A30	Model A40
4	20	30	30	—	—	—
8	40	60	60	75	—	—
12	—	70	70	80	—	—
16	—	—	75	90	100	—
20	—	—	80	95	110	—
24	—	—	85	100	120	—
28	—	—	90	105	130	—
32	—	—	95	110	140	160
36	—	—	100	—	150	—
40	—	—	—	115	160	—
48	—	—	—	—	170	240
64	—	—	—	—	—	320
80	—	—	—	—	—	380
96	—	—	—	—	—	425

Figure 5-4. Job Space (K)

For each line, protocol, line type, and controller, the machine pool must include some additional main storage. Figure 5-5 is given as a simple guide to the additional storage required. These values are based on estimates of the number and types of communications lines that may appear for each model and main storage size. However, you are able to do a better job because you know the communications configuration of the system. To estimate this:

1. Add 125K for each line.
2. Add 100K for each protocol (BSC, SDLC, ASYNC, and so on).
3. Add 25K for each controller.

Main Storage Size (M)	Model A4	Model A6	Model A10	Model A20	Model A30	Model A40
4	250	250	250	—	—	—
8	350	350	350	400	—	—
12	—	475	475	525	—	—
16	—	—	550	675	750	—
20	—	—	650	775	825	—
24	—	—	750	850	900	—
28	—	—	900	950	1000	—
32	—	—	1025	1075	1125	1250
36	—	—	1200	—	1250	—
40	—	—	—	1350	1375	—
48	—	—	—	—	1500	1750
64	—	—	—	—	—	2200
80	—	—	—	—	—	2800
96	—	—	—	—	—	3250

Figure 5-5. Communications Space (K)

Lastly, certain system functions, when active, require additional space in the machine pool. If you plan to use any of the functions shown in Figure 5-6 you should add the appropriate amount of main storage to the machine pool. The IPL tuner determines the functions used from the machine configuration information.

Function	Addition to Machine Pool
3270 Emulation or Remote Attach	50K
Check sum	5% of main storage size
Save/Restore	32K
DBCS	50K
X.25	48K
S/36 Environment	350K
Token-Ring Local Area Network	250K

Figure 5-6. Functional Space

Choosing Your Pool Configuration

After you have set your machine pool size, you need to decide what to do with the remaining storage. Your basic choices are:

- Do not create any additional pools.
- Create a separate pool for QINTER and QSPL.

If you choose to create separate pools for QINTER and QSPL, you should use Figure 5-7 to determine the QSPL pool size and activity levels. Then use Figure 5-8 on page 5-18 to calculate the *BASE pool size and activity levels. The remaining storage will be the QINTER pool size. In order to calculate an activity level for the QINTER pool, divide the pool size by the value determined from Figure 5-9 on page 5-18. The IPL tuner uses the same tables to establish pool sizes and activity levels.

# Writers	Size (K)	Activity Level
1	40	1
2	70	2
3	100	3
4	120	3
5	140	3
>5	150	3

Figure 5-7. QSPL Pool Sizes and Activity Levels

Figure 5-8 assumes that interactive work is running at the same time as many batch jobs.

Main Storage Size (M)	Model A4 Pool Size (K)/ Act Lvl	Model A6 Pool Size (K)/ Act Lvl	Model A10 Pool Size (K)/ Act Lvl	Model A20 Pool Size (K)/ Act Lvl	Model A30 Pool Size (K)/ Act Lvl	Model A40 Pool Size (K)/ Act Lvl
4	500 / 2	500 / 2	500 / 2	—	—	—
8	1000 / 3	1000 / 3	1000 / 3	1000 / 3	—	—
12	—	1250 / 3	1250 / 3	1250 / 3	—	—
16	—	—	1500 / 3	1500 / 4	1500 / 4	—
20	—	—	1750 / 3	1750 / 4	1750 / 4	—
24	—	—	2000 / 3	2000 / 5	2000 / 5	—
28	—	—	2000 / 3	2350 / 5	2350 / 5	—
32	—	—	2000 / 3	2700 / 5	2750 / 6	2750 / 6
36	—	—	2000 / 3	—	3000 / 6	—
40	—	—	—	3500 / 5	3500 / 6	—
48	—	—	—	—	4000 / 6	4000 / 6
64	—	—	—	—	—	5250 / 7
80	—		—	—	—	6500 / 7
96	—	—	—	—	—	8000 / 7

* Pool separation is not possible.

Figure 5-8. *BASE Pool Sizes and Activity Levels

Main Storage Size (M)	Activity Level Factor (K)
≤ 12	300K
16 — 28	450K
32 — 48	600K
64 — 96	900K

Figure 5-9. QINTER Activity Level Factor

Adjustments to Pool Sizes and Activity Levels

Once you have set your initial pool sizes and activity levels, you should begin to observe your system performance. When you are observing performance:

- Use both WRKSYSSTS and WRKACTJOB.
- Observation intervals should be 5 minutes.
- The system should be handling *normal* workloads. Observations during noon hour may not give meaningful data.
- Apply the page fault rate guidelines shown earlier in this chapter.

If your observations indicate page fault rates outside the stated guidelines, you should determine the cause of the high page fault rate. Do not make adjustments to tuning if some abnormal condition is causing the high page fault rate. Interactive program compiles, communications ERP, OPNQRYF, application errors, and sign-off activity are examples of irregular activities which may cause a severe hit to performance. Rather than adjusting the system tuning, you should determine the reasons for these activities and try to remove as many as possible from your operating environment.

If you have decided that the page fault rates in your system are due to *natural causes*, you should take action to adjust your tuning parameters. You will be dealing primarily with pool size and activity levels. Also, you should begin by tuning the machine pool first. As mentioned earlier, you can only reduce machine pool page fault by increasing the size of the pool. Once you have reached the values shown earlier as *good*, you can stop.

If your machine pool is experiencing very low page fault rates (<0.4), you should decrease the size of the machine pool. If the page fault rate is this low, you may be affecting work in some other pools.

After the machine pool fault rate has been tuned, you now need to focus on the other pools in the system. Once again, we have two choices for adjusting these pools: size and activity level.

Adjusting Activity Levels

Adjusting the activity level may be the most beneficial way to improve the tuning in pools 2 through 16. The possible conditions that can exist in these pools are:

- Low fault rate (<15) with a high number of wait-to-ineligible transitions
- A high fault rate (>25) with either a high or low number of wait-to-ineligible transitions
- A moderate fault rate (15 — 25) with a moderate number of wait-to-ineligible transitions

In the first situation, the activity level is probably set too low. The paging rate is low because only a few jobs are allowed in the pool at the same time. As a result, the jobs must wait for an activity level before they can run. Increase the activity level by one until the paging rate in the pool reaches the acceptable level. Be certain to measure and compare each change so that you can identify that best activity level.

In the second situation, the activity level is probably set too high. In this case, too many jobs are allowed into the pool at the same time. Instead of doing productive work in main storage, the jobs must read information to complete their transactions many times. Demand for space in the pool is so high that jobs are unable to use their pages before another job has overlaid their information. This condition is called thrashing. In order to reduce thrashing, reduce the activity level by one until paging returns to an acceptable rate. Once again, it is necessary to measure and observe performance after each change.

The last situation indicates that the activity level may be set right, but the pool size is too small. In order to alleviate this case, more storage and, possibly, more activity levels should be configured for this pool. This is explained below.

As you adjust the activity level, you may find that you are cycling through the three cases described above. Using your WRKACTJOB command data, select the tuning values that have provided the best performance. If performance is still inadequate, evaluate the processing unit and disk resource utilizations.

Adjusting Pool Sizes

If you have set your activity level at its optimum level, examine the sizes of the pools in your system. If you detect a pool with a low fault rate and a low wait-to-ineligible count, the pool probably has too much storage. If there are pools that are still classified as the last situation (from above), reduce the size of storage-rich pools and add it to the pools with less storage. Reductions should be done in decrements of 10%. Once again, each change should be measured.

Reviewing Your Performance

Once you have set good tuning values, you should periodically review them to ensure your system continues to do well. If performance is not satisfactory in spite of your best efforts, you should evaluate the capabilities of your configuration. In order to meet your objectives, you may need to consider the following:

- Processor upgrades
- Additional storage devices and/or controllers
- Additional main storage
- Application modification

By applying one or more of these approaches, you may achieve your objectives. If, after a reasonable effort, you are still unable to meet your objectives, you should determine if your objectives are realistic for the type of work you are doing.

Specialized Tuning

After you have had some time to observe your system performance, you may want to consider more specialized tuning for your environment. Some considerations are:

- Use PURGE(*NO) for interactive jobs.
- Separate batch work from *BASE.
- Use multiple pools for interactive jobs.
- Use multiple pools for batch jobs.

When applying these tuning techniques, it is advisable to change the value of QPFRADJ to 0. Otherwise, system performance following an IPL may not be what you expect.

Specifying PURGE(*NO) for Interactive Jobs

As discussed earlier, the characteristics of interactive jobs are different when PURGE(*NO) is specified on the job class rather than PURGE(*YES). When you specify PURGE(*YES), the system performs some *automatic* disk I/O operations whenever the job enters and leaves an activity level. These operations may not always be necessary. PURGE(*NO) performs disk I/O operations only when necessary. In environments with sufficient main storage, PURGE(*NO) should be considered.

To determine if your interactive jobs are candidates for PURGE(*NO), your observations should show the following:

- Page fault rate in QINTER pool of less than 15
- Few or no wait-to-ineligible transitions

Then, multiply the number of jobs by 125K. If the result of your multiplication is approximately equal to your pool size, set the PURGE attribute to *NO.

When you change to PURGE(*NO), you are usually giving better performance to steady users. Casual users may not benefit in the same manner. Since a casual user does not continually enter transactions, the objects used by the job will usually be removed from main storage between transactions. If the job's objects have been removed, getting them back to main storage often results in slightly longer response time when the PURGE attribute is *NO. For this situation, you may want to set multiple interactive pools.

Separate Batch Work from *BASE

To this point, batch jobs have shared *BASE storage with system jobs (for example, SCPF, QSYSARB, and subsystem monitors) and system transients (for example, file OPEN/CLOSE). As a result, batch jobs compete for space in the *BASE pool. Batch performance may be improved if the jobs are moved to a separate pool. The size of the pool should be 300K for each job running at the same time. The activity level should be set equal to the number of jobs running at the same time. You should separate batch jobs only if batch is continually active. Once you have separated batch from *BASE, the storage will not be used by anyone in the system when there are no batch jobs running. This will not make efficient use of main storage.

Multiple Pools for Interactive Jobs

As mentioned above, there are instances where distinct sets of interactive users should be isolated. Some examples are:

- Programmers
- Users performing office-type functions exclusively
- Data entry

In the first two cases, you are attempting to isolate users who are performing the same functions. Often, the functions performed by these users are different from the functions used by all other users. Also, some of these users may be classified as casual users and isolating them will help protect their objects. In the third case, you are isolating extremely active users to give the best possible response time for their activity.

If you are considering this approach, you need to determine how many users will be put in each pool. Then, after setting the necessary routing entries, you need to calculate a pool size, activity level, and PURGE attribute for each pool. If you are setting a separate pool for casual users:

- Set the PURGE attribute to *YES.
- To calculate the activity level, divide the number of work stations to be routed to the pool by 4.
- Multiply the activity level by 300K to set the pool size.

Remember, once you have separated your work, you no longer have the capability of sharing the storage. Also, each pool should be tuned to the optimal pool size and activity level.

Multiple Pools for Batch Jobs

Each batch job may use objects that are different from the objects used by other batch jobs in the same pool. Production batch does not use anything that is the same as a program compile. Long-running jobs may not perform well if sharing a pool with short-running jobs. S36E evokes may disrupt other batch jobs' performance. If you feel you have some of these situations, you may want to set up multiple batch pools. When you separate the pools, set up a pool for each batch job type and use 300K for each job in the pool. Each pool should have an activity level of 1 and only one active job. Again, if there is no work being performed in these pools, the main storage will not be used by the system to support jobs in other pools.

Summary

The system allows you many options for tuning your system. The concepts presented here give you some general guidelines, not all the answers. Each system environment is unique and requires you to observe performance and make adjustments that are best for your environment.

Chapter 6. Collecting Performance Data

This chapter explains how to collect performance data, and how the collection occurs. It also contains reference information that shows the format and content of the data base files collected. This information is provided so the user can collect performance data collection and write an application to analyze the data.

Why to Collect Performance Data

You may choose to collect performance data for any of the following reasons:

- To do analysis with the IBM Performance Analysis licensed program
- To do analysis on another system that has the IBM Performance Analysis licensed program installed
- To gather input for an analysis application that reduces the data into meaningful reports

Preparing to Collect Performance Data

Before starting data collection, consider what type of data to collect, how often to collect it, and when to collect it.

Data collection may slightly degrade performance of the system due to the time involved in collecting the performance data. The amount of degradation is related to the amount of data collected, so you should only collect the data you need.

Three types of data can be collected:

- System data. This data consists of performance data relating to the following: all jobs on the system, devices attached to the system, storage pools, communications I/O processors, disk I/O processors, local work station I/O processors, and work station response times.
- Communications data. This data includes all of the previously listed system data and statistics for the following protocols: binary synchronous, asynchronous, X.25, token-ring network, and synchronous data link control (SDLC).
- Trace data. This data includes internal system trace data from the vertical microcode trace table. This type of data should not be collected unless it is going to be analyzed by the IBM Performance Analysis licensed program.

How to Collect Performance Data

This section explains how to start and end the collection of performance data.

Starting Performance Data Collection

The Start Performance Monitor (STRPFRMON) command starts performance data collection. Issuing this command submits a batch job to a job queue. When the performance monitor batch job is started from the job queue, data collection begins. The job continues to collect data until one of the following occurs:

- The time limit specified for data collection is reached.
- The End Performance Monitor (ENDPFRMON) command is issued.
- The performance monitor job is ended abnormally (End Job (ENDJOB), End Subsystem (ENDSBS), or End System (ENDSYS) command).

While the performance monitor job is active, the monitor periodically collects data and puts it in the performance data base files. For more information on how data collection occurs, see “How Performance Data Collection Occurs” on page 6-4. For more information on the Start Performance Monitor (STRPFRMON) command, see the *CL Reference*.

Collecting System Data Only

To collect only system data, specify *SYS for the DATA parameter on the STRPFRMON command.

Collecting System and Communications Data

To collect communications data, specify *ALL for the DATA parameter on the STRPFRMON command.

Note: Communications data is collected in addition to system data, not instead of it.

Collecting Trace Data

To collect trace data, you must specify a value other than *NONE for the TRACE parameter on the STRPFRMON command. This causes the performance monitor to start the trace (using the Trace Internal (TRCINT) command). Turning on the trace causes the system to log various activities into an internal trace table. Eventually, the trace data is dumped from the internal trace table into a data base file (QAPMDMPT).

Dumping the trace data from the internal trace table into the trace data base file also degrades performance. It is probably not a good idea to dump the trace during peak activity on a heavily loaded system. So when the TRACE parameter indicates to start the trace, you should also consider when the trace should be dumped to the data base file.

The dump trace (DMPTRC) parameter on the STRPFRMON command allows you to specify if the dump should take place when the performance monitor ends. The DMPTRC parameter is also on the ENDPFRMON command, so that you can override what you specified for the DMPTRC parameter on the STRPFRMON command.

If the trace table is not dumped when the performance monitor ends, it can be dumped later with the Dump Trace (DMPTRC) command. This command allows dumping of the trace table (which may degrade system performance on heavily loaded systems) to be delayed until a time that would not affect other users. If the trace is not dumped when the performance monitor ends, the trace data remains in the internal system trace table. If the trace table is cleared for any reason (running another system trace clears the trace table), the performance trace data is lost. Therefore, it is a good idea to dump the trace table as soon as possible after the performance monitor ends.

Dumping Trace Data

The DMPTRC command is used to dump information from an internal trace table into a data base file. This command allows the dump to be done by the job that issued the command, or, by submitting a batch job to a job queue, to have the trace dumped by the batch job.

For more information on the DMPTRC command, see the *CL Reference*.

The Internal Trace Table

The internal trace table has a maximum size of 16M of storage. Before starting any traces, the performance monitor sets the trace table size *limit* to this maximum value to allow the most information possible to be collected. The trace table can use up to 16M, but is not initially set to that value. If the trace is not started, the size of the trace table is not changed.

After the performance monitor starts the trace, the system continues to log trace data until either the trace table fills up or the trace stops (which is done automatically when the performance monitor ends). If the trace table fills up while the performance monitor is running (and the performance monitor started the system trace), a message is sent to the system operator's message queue and a user's message queue (if one was specified on the MSGQ parameter of the STRPFRMON command). The performance monitor then automatically turns off the trace. At that point, three options are available:

- Immediately dump the trace table to a data base file (by using the DMPTRC command). Using this option ensures that the trace data is not written over if other traces are turned on. However, this approach may degrade performance on heavily loaded systems.
- Wait until the performance monitor ends and have the performance monitor dump the trace at that time (this is the default). If another trace is started before the performance monitor ends, the data in the internal trace table is written over, losing the trace information that the performance monitor produced.
- Do not dump the trace when the performance monitor ends. Instead, dump the trace at a convenient time with the DMPTRC command.

Ending Performance Data Collection

The ENDPFRMON command is used to end performance data collection. Issuing the command causes the performance monitor batch job to make a final collection of performance data and then end itself. This command should be used instead of the ENDJOB, ENDSBS, and ENDSYS commands if you want the final data collected. Only the ENDPFRMON command causes the final data to be collected.

For more information on the ENDPFRMON command, see the *CL Reference*.

How Performance Data Collection Occurs

Performance data collection occurs when a batch job is submitted to a job queue with the STRPFRMON command. No data is collected until the job is started from the job queue, so an active job queue should be specified for the JOBQ parameter of the STRPFRMON command.

Although the performance monitor is a batch job, it runs at a higher priority than interactive jobs, but lower than system-level jobs.

After the performance monitor job is started, it opens the data collection data base files and performs other initialization functions. After initialization, the performance monitor job waits for a data collection interval to occur.

When the performance monitor collects data, it retrieves performance counters from the hardware devices attached to the system. The counters that are stored in the data base file for an interval indicate the amount of resources used for just that interval (not a summary of the total use since the performance monitor started).

The performance monitor job collects data for the amount of time specified on the STRPFRMON command. When the time limit is reached, the performance monitor puts the final performance counters into the data base file (regardless of whether a complete interval occurred) and then ends itself. This means that if you specified a 60-minute interval and a 2-1/2 hour time limit, the data base files contain entries placed in the files at 1 hour, 2 hours, and 2-1/2 hours.

If the performance monitor was started and you want to end it before the time limit is reached, you can use the ENDPFRMON command. The ENDPFRMON command puts the final performance counters into the data base files, and ends the performance monitor batch job (and, if specified, dumps the performance trace into a data base file).

Internal Data Collection Intervals

A data collection interval is the amount of time during which the performance monitor job is active to retrieve data from the system.

Although you can specify that data is to be placed in the data base files at intervals ranging from 5 to 60 minutes (the default is 15 minutes), counter limitations require the performance monitor to retrieve performance data from the system every 5 minutes. Some devices have counters that can wrap (when the counter reaches its maximum value, it is reset to zero). The performance monitor can handle one wrap when calculating the counts for the interval, but if the counter wraps twice before the performance monitor retrieves the counter, the data is lost for one wrap (there is no indication that the counter wrapped twice). The maximum amount of time the performance monitor can run before risking a double wrap is 5 minutes. Therefore, the performance monitor must retrieve the data every 5 minutes to ensure that there is no loss of data.

When the Data Is Collected

Regardless of the value specified for the collection interval, the performance monitor must collect counters from the devices attached to the system every five minutes. Any of these five-minute intervals that do not coincide with the user specified collection interval are called an *internal interval.* Any of the five-minute intervals that coincide with the user-specified collection intervals are called *data base intervals,* because the performance monitor writes the counters collected for each user-specified interval to the performance data base files.

The following example illustrates the difference between an internal interval and a data base interval (assume that 15 minutes was specified for the INTERVAL parameter for the STRPFRMON command):

```
                               Data Base Interval            Data Base Interval
           |                             |                             |
           |                             |                             |
           |           Internal Interval |           Internal Interval |
           |                   |         |                   |         |
           | Internal Interval |         | Internal Interval |         |
. Time in  | . . . . | . . . . | . . . . | . . . . | . . . . | . . . . | . .
  Minutes  0         5        10        15        20        25        30
|                                                                          |
<--------------------------- Collection Period ---------------------------->
```

During internal intervals, the only data that is collected is from the devices attached to the system. This data is saved in internal tables until a data base interval occurs, when it is written to the data base files.

During data base intervals, all of the data specified for collection is retrieved from the devices and the system, the values for the data base interval are calculated (remember each interval shows the amount of use for just that interval), and the counters are written to the performance data base files.

It is also possible for the performance monitor to become active between intervals. Certain events on the system cause the performance monitor to be notified when they occur. When one of these events occurs, the performance monitor momentarily becomes active (usually just long enough to store information about the event in an internal table), and then waits for the next collection interval to occur.

One of these events occurs when a job ends on the system. The only time that the performance monitor puts data into the data base file other than at a data base interval is when a job ends on the system (an entry is then inserted for that job for the final amount of resources used since the last data base interval).

Notification of Performance Monitor Status

While the performance monitor is active, it sends information messages to the system operator's message queue. These messages notify the system operator when the status of the performance monitor changes or an error occurs. Typical messages sent indicate that:

- The performance monitor was started (from the job queue).
- The performance monitor encountered an unexpected error.
- Another user issued the End Performance Monitor (ENDPFRMON) command.
- The performance monitor has ended.

Specifying the MSGQ parameter on the STRPFRMON command causes the performance monitor to send messages to the user's message queue whenever it sends messages to the system operator's message queue.

Data Base File Management

The performance monitor uses many different data base files while collecting performance data, but the performance monitor places the data into the same member for all the files it uses. If the member specified on the STRPFRMON command does not exist when the performance monitor starts, it adds a member by that name to all the files it uses.

Because the default for the member name (MBR) parameter for the STRPFRMON command is *GEN (create the member name), it is possible to quickly add many members to the performance monitor data base files. When using the default, you should periodically delete members in the performance data base files when they are no longer used.

An alternative to deleting members is to delete the files themselves. If the performance monitor does not find a file in the specified library, it creates the file in that library and adds the specified member to the file.

Another alternative is to specify several member names over and over (if running every day of the week, name the members after the days of the week).

Estimating File Size for Collecting Performance Data

This section shows how to estimate the file size needed to collect system data and to collect system and communications data.

Collecting System Data Only: To estimate the amount of disk space for collecting only system data using the STRPFRMON command and specifying DATA(*SYS), add the following:

55 plus the product of the number of intervals times the sum of:

1059	System data per interval
261	Times the average number of jobs active in the interval
157	Times the number of disk arms
72	Times the number of pools
50	Times the average number of local work stations active in the interval

For example,

```
Disk space = 55 + {#intervals x [(1059) + (261 x avg #jobs) +
(157 x #disk arms) + (72 x #pools) + (50 x #ws)]}
```

Collecting System and Communications Data: To estimate the amount of disk space for collecting both system and communications data using the STRPFRMON command by specifying DATA(*ALL), add the following:

55 plus the product of the number of intervals times the sum of:

1059	System data per interval
261	Times the average number of jobs active in the interval
157	Times the number of disk actuators
72	Times the number of storage pools
50	Times the average number of local work stations active in the interval
157	Times the average number of SDLC lines active in the interval
73	Times the average number of asynchronous lines active in the interval
199	Times the average number of binary synchronous communications (BSC) lines active in the interval
327	Times the average number of X.25 lines active in the interval
220	Times the average number of token-ring network (TRLAN) lines active in the interval
116	Times the average number of communications controllers active in the interval
520	Times the average number of storage device controllers
133	Times the average number of twinaxial work station controllers active in the interval

Content of Performance Data Files

The tables on the following pages show the format and content of performance data files collected by the system when using data collection commands. They also describe some of the relationships between the counters. The file names and descriptions used in the tables as follows:

- **Configuration Data (collected once per session):**

File Name	Description
QAPMCONF	System configuration data

- **Performance Data (collected each interval):**

File Name	Description
QAPMSYS	System performance data
QAPMJOBS	Job data (one per job)
QAPMDISK	Disk storage data (one per read/write head)
QAPMPOOL	Main storage data (one per system storage pool)
QAPMHDLC	HDLC statistics (one per link)
QAPMASYN	Asynchronous statistics (one per link)
QAPMBSC	Binary synchronous statistics (one per link)
QAPMX25	X.25 statistics (one per link)
QAPMECL	Token-ring local area network statistics (one per link)
QAPMCIOP	Communications controller data (one per controller)
QAPMDIOP	Storage device controller data (one per controller)
QAPMLIOP	Twinaxial work station controller data (one per controller)
QAPMRESP	Local work station response time (one per work station)

- **Trace Data:**

File Name	Description
QAPMDMPT	System trace data (general description – no field and byte detail)

All the data except for QAPMCONF is collected for each sample. QAPMCONF contains system configuration information that is reported only at the beginning of the performance monitor data collection job.

Terms Used

The following abbreviations are used in the following tables.

C = character in the *Attributes* column.

PD = packed decimal in the *Attributes* column.

IOP = input/output processor or I/O processor. The processors that control the activity between the host system and other devices such as disks, display stations, and communications lines.

DCE = data-circuit-terminating equipment.

MAC = medium-access control. An entity in the communications IOP.

LLC = logical link control. An entity in the communications IOP.

Beacon frame = a frame that is sent when the ring is inoperable.

File Name: QAPMCONF

System Configuration File Entries Figure 6-1 lists the various fields in the configuration file entries.:

Field Name	Description	Attributes
GIOP	IOP BUS number/address (see note 1)	C (1)
GUNIT	Device address (see note 2)	C (2)
GLVL	Resource level (see note 3)	C (1)
GKEY	See note 4	C (1)
GDES	See note 4	C (10)

Figure 6-1. System Configuration Data

Notes:

1. GIOP is a value only when the containing record is a configuration record. The first 3 bits are the bus number (value 0–2) and the remaining 5 bits are the bus address (value 0–31).
2. GUNIT is a value only when the containing record is a configuration record.
3. GLVL =

   1 – Resource is an IOP
   2 – Resource is an I/O adapter or controller
   3 – Resource is a port or device

   GLVL is blank when the previous two fields (GIOP and GUNIT) are blank.
4.

GKEY	GDES
1	Performance monitor start date (yy/mm/dd).
2	Performance monitor start time (hh:mm:ss).
3	Model number (character 4 followed by 6 blanks).
4	Main storage size in K-bytes (character 7).
5	Communications data collected (Y/N).
6	Machine serial number (character 8).
7	First response time boundary (PD (3,0)). Lower boundary of the first response time monitor bracket.
8	Second response time boundary (PD (3,0)). Lower boundary of the second response time monitor bracket.
9	Third response time boundary (PD (3,0)). Lower boundary of the third response time monitor bracket.
10	Fourth response time boundary (PD (3,0)). Lower boundary of the fourth response time monitor bracket and upper boundary of the last response time monitor bracket.
11	System ASP capacity in bytes (PD (15,0)). Total number of bytes of auxiliary storage allocated to the system ASP for the storage of data.
12	Checksum protection on (Y/N).

L	Resource name of the local work station controller.
C	Resource name of the communications IOP.
D	Resource name of the storage IOP.

File Name: QAPMSYS

System Interval File Entries The following terms are used in Figure 6-2, and are repeated for each group of jobs:

- Number of data base read operations. Total number of physical read operations for data base functions.
- Number of non-data-base read operations. Total number of physical read operations for non-data-base functions.
- Number of write operations. Total number of physical write operations.
- Number of print lines. Number of lines written by the program. This number does not reflect what is actually printed. Spooled files can be ended, or printed with multiple copies.
- Number of data base writes/reads (logical). Number of times the data base module was called. This number does not include I/O operations to readers/writers, or I/O operations caused by the Copy Spooled File (CPYSPLF) or Display Spooled File (DSPSPLF) command. If SEQONLY(*YES) is in effect, these numbers show each block of records read, not the number of individual records read.
- Number of communications writes/reads (logical). These do not include remote work station activity. They include only activity related to OS/400-ICF files when the I/O is for a communications device.
- Number of times PAG was brought. Total number of times the program access group (PAG) was brought into main storage after it was purged when running with PURGE(*YES) parameter.
- Number of times PAG was purged. Total number of times the PAG was purged when the job entered a long wait state and was running with PURGE(*YES) option.

Note: Blocked I/O is considered one data base transaction.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
SYSCPU	Total system processing unit time used (in milliseconds).	PD (9,0)
SYDPGF	Directory page faults: Number of times a page of the auxiliary storage directory was transferred to main storage for a look-up or an allocation operation.	PD (11,0)
SYAPGF	Access group member page faults: Number of times a page of an object contained in an access group was transferred to main storage independently of the access group. This transfer occurs when the containing access group was purged, or because portions of the containing access group are displaced from main storage.	PD (11,0)
SYMPGF	Microcode page faults: Number of times a page of microcode was transferred to main storage.	PD (11,0)
SYMCTR	Microtask read operations: Number of transfers of one or more pages of data from auxiliary storage because of a microtask rather than a process.	PD (11,0)
SYMCTW	Microtask write operations: Number of transfers of one or more pages of data from main storage to auxiliary storage because of a microtask rather than a process.	PD (11,0)
SYSASP	System auxiliary storage pools space available: Number of bytes of space on auxiliary storage available for allocation in the system ASP that is not currently assigned to machine interface (MI) objects or internal machine functions.	PD (15,0)
SYPRMW	Permanent data transferred from main storage: Number of 512-byte blocks of permanent data transferred from main storage to the system ASP in auxiliary storage since the last sample.	PD (11,0)
SYXSRW	Redundancy data transferred from main storage: Number of 512-byte blocks of redundant data transferred from main storage to the system ASP in auxiliary storage since the last sample.	PD (11,0)
SYEAOT	Total number of effective address overflow exceptions.	PD (11,0)
SYEAOL	Total number of effective address-length overflow exceptions.	PD (11,0)
SYBSYC	Busy count: Total number of busy exceptions.	PD (11,0)
SYSIZC	Size count: Total number of size exceptions.	PD (11,0)
SYDECD	Decimal data count: Total number of decimal data exceptions.	PD (11,0)
SYSEZC	Seize count: Total number of seize wait exceptions.	PD (11,0)
SYSYNL	Synchronous lock conflict count.	PD (11,0)
SYASYL	Asynchronous lock conflict count.	PD (11,0)

Figure 6-2 (Part 1 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SYVFYC	Verify count.	PD (11,0)
SYAUTH	Authority look-up count.	PD (11,0)
SYCHNB	Channel busy.	PD (11,0)
	Reserved.	C (30)
SYLRT1	Transactions in first response time monitor bracket: Total number of local work station transactions with response time less than the value of boundary 1 specified on the STRPFRMON command.	PD (9,0)
SYLRT2	Transactions in second response time monitor bracket: Total number of local work station transactions with response time less than the value of boundary 2 and greater than the value of boundary 1 specified on the STRPFRMON command.	PD (9,0)
SYLRT3	Transactions in third response time monitor bracket: Total number of local work station transactions with response time less than the value of boundary 3 and greater than the value of boundary 2 specified on the STRPFRMON command.	PD (9,0)
SYLRT4	Transactions in fourth response time monitor bracket: Total number of local work station transactions with response time less than the value of boundary 4 and greater than the value of boundary 3 specified on the STRPFRMON command.	PD (9,0)
SYLRT5	Transactions in fifth response time monitor bracket: Total number of local work station transactions with response time greater than the value of boundary 5 specified on STRPFRMON command.	PD (9,0)
SDCPU	Total processing unit time used (in milliseconds) by target distributed data management (DDM) job.	PD (11,0)
SDRES1	Reserved.	PD (11,0)
SDRES2	Reserved.	PD (11,0)
SDPRTL	Total number of print lines of all target DDM jobs.	PD (11,0)
SDPRTP	Total number of print pages of all target DDM jobs.	PD (11,0)
SDSPD	Total count of suspended time of target DDM jobs.	PD (11,0)
SDRRT	Total count of time a target DDM job waited during rerouting.	PD (11,0)
SDNEW	Number of new target DDM job.	PD (11,0)
SDTERM	Number of ended target DDM jobs.	PD (11,0)
SDPDBR	Total number of physical data base reads by target DDM jobs.	PD (11,0)
SDPNDB	Total number of physical non-data-base reads by target DDM jobs.	PD (11,0)
SDPWRT	Total number of physical writes by target DDM jobs.	PD (11,0)
SDLDBR	Total number of logical data base reads by target DDM jobs.	PD (11,0)
SDLDBW	Total number of logical data base writes by target DDM jobs.	PD (11,0)

Figure 6-2 (Part 2 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SDLDBU	Total number of miscellaneous data base operations by target DDM jobs.	PD (11,0)
SDCMPT	Total number of communications writes by target DDM jobs.	PD (11,0)
SDCMGT	Total number of communications reads by target DDM jobs.	PD (11,0)
SDBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SDPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SWCPU	Total processing unit time (in milliseconds) used by PC Support applications.	PD (11,0)
SWRES1	Reserved.	PD (11,0)
SWRES2	Reserved.	PD (11,0)
SWPRTL	Total number of print lines of all PC Support application jobs.	PD (11,0)
SWPRTP	Total number of print pages of all PC Support application jobs.	PD (11,0)
SWSPD	Total time PC Support application jobs were suspended.	PD (11,0)
SWRRT	Total time a PC Support applications job waited during rerouting.	PD (11,0)
SWNEW	Number of started PC Support application jobs.	PD (11,0)
SWTERM	Number of ended PC Support application jobs.	PD (11,0)
SWPDBR	Total number of physical data base reads by PC Support application jobs.	PD (11,0)
SWPNDB	Total number of physical non-data-base reads by PC Support application jobs.	PD (11,0)
SWPWRT	Total number of physical writes by PC Support application jobs.	PD (11,0)
SWLDBR	Total number of logical data base reads by PC Support application jobs.	PD (11,0)
SWLDBW	Total number of logical data base writes by PC Support application jobs.	PD (11,0)
SWLDBU	Total number of miscellaneous data base operations by PC Support application jobs.	PD (11,0)
SWCMPT	Total number of communications writes by PC Support application jobs.	PD (11,0)
SWCMGT	Total number of communications reads by PC Support application jobs.	PD (11,0)
SWBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SWPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SPCPU	Total processing unit time (in milliseconds) used by pass-through target jobs.	PD (11,0)
SPRES1	Reserved.	PD (11,0)
SPRES2	Reserved.	PD (11,0)
SPPRTL	Total number of print lines of all pass-through target jobs.	PD (11,0)
SPPRTP	Total number of print pages of all pass-through target jobs.	PD (11,0)
SPSPD	Total count of suspended time of pass-through target jobs.	PD (11,0)
SPRRT	Total count of time a pass-through target job waited during rerouting.	PD (11,0)
SPNEW	Number of started pass-through target jobs.	PD (11,0)

Figure 6-2 (Part 3 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SPTERM	Number of ended pass-through target jobs.	PD (11,0)
SPPDBR	Total number of physical data base reads by pass-through target jobs.	PD (11,0)
SPPNDB	Total number of physical non-data-base reads by pass-through target jobs.	PD (11,0)
SPPWRT	Total number of physical writes by pass-through target jobs.	PD (11,0)
SPLDBR	Total number of logical data base reads by pass-through target jobs.	PD (11,0)
SPLDBW	Total number of logical data base writes by pass-through target jobs.	PD (11,0)
SPLDBU	Total number of miscellaneous data base operations by pass-through target jobs.	PD (11,0)
SPCMPT	Total number of communications writes by pass-through target jobs.	PD (11,0)
SPCMGT	Total number of communications reads by pass-through target jobs.	PD (11,0)
SPBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SPPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SMCPU	Total processing unit time (in milliseconds) used by MRT jobs (System/36 environment only).	PD (11,0)
SMRES1	Reserved.	PD (11,0)
SMRES2	Reserved.	PD (11,0)
SMPRTL	Total number of print lines of all MRT jobs (System/36 environment only).	PD (11,0)
SMPRTP	Total number of print pages of all MRT jobs (System/36 environment only).	PD (11,0)
SMSPD	Total time MRT jobs (System/36 environment only) were suspended.	PD (11,0)
SMRRT	Total time a MRT job (System/36 environment only) waited during rerouting.	PD (11,0)
SMNEW	Number of started MRT jobs (System/36 environment only).	PD (11,0)
SMTERM	Number of ended MRT jobs (System/36 environment only).	PD (11,0)
SMPDBR	Total number of physical data base reads by MRT jobs (System/36 environment only).	PD (11,0)
SMPNDB	Total number of physical non-data-base reads by MRT jobs (System/36 environment only).	PD (11,0)
SMPWRT	Total number of physical writes by MRT jobs (System/36 environment only).	PD (11,0)
SMLDBR	Total number of logical data base reads by MRT jobs (System/36 environment only).	PD (11,0)
SMLDBW	Total number of logical data base writes by MRT jobs (System/36 environment only).	PD (11,0)
SMLDBU	Total number of miscellaneous data base operations by MRT jobs (System/36 environment only).	PD (11,0)
SMCMPT	Total number of communications writes by MRT jobs (System/36 environment only).	PD (11,0)
SMCMGT	Total number of communications reads by MRT jobs (System/36 environment only).	PD (11,0)

Figure 6-2 (Part 4 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SMBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SMPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
S6CPU	Total processing unit time (in milliseconds) used by System/36 environment jobs.	PD (11,0)
S6RES1	Reserved.	PD (11,0)
S6RES2	Reserved.	PD (11,0)
S6PRTL	Total number of print lines of all System/36 environment jobs.	PD (11,0)
S6PRTP	Total number of print pages of all System/36 environment jobs.	PD (11,0)
S6SPD	Total time System/36 environment jobs were suspended.	PD (11,0)
S6RRT	Total time a System/36 environment job waited during rerouting.	PD (11,0)
S6NEW	Number of started System/36 environment jobs.	PD (11,0)
S6TERM	Number of ended System/36 environment jobs.	PD (11,0)
S6PDBR	Total number of physical data base reads by System/36 environment jobs.	PD (11,0)
S6PNDB	Total number of physical non-data-base reads by System/36 environment jobs.	PD (11,0)
S6PWRT	Total number of physical writes by System/36 environment jobs.	PD (11,0)
S6LDBR	Total number of logical data base reads by System/36 environment jobs.	PD (11,0)
S6LDBW	Total number of logical data base writes by System/36 environment jobs.	PD (11,0)
S6LDBU	Total number of miscellaneous data base operations by System/36 environment jobs.	PD (11,0)
S6CMPT	Total number of communications writes by System/36 environment jobs.	PD (11,0)
S6CMGT	Total number of communications reads by System/36 environment jobs.	PD (11,0)
SECPU	Total processing unit time (in milliseconds) used by communications batch jobs.	PD (11,0)
SERES1	Reserved.	PD (11,0)
SERES2	Reserved.	PD (11,0)
SEPRTL	Total number of print lines of all communications batch jobs.	PD (11,0)
SESPD	Total time communications batch jobs were suspended.	PD (11,0)
SERRT	Total time a communications batch job waited during rerouting.	PD (11,0)
SENEW	Number of started communications batch jobs.	PD (11,0)
SETERM	Number of ended communications batch jobs.	PD (11,0)
SEPDBR	Total number of physical data base reads by communications batch jobs.	PD (11,0)
SEPNDB	Total number of physical non-data-base reads by communications batch jobs.	PD (11,0)
SEPWRT	Total number of physical writes by communications batch jobs.	PD (11,0)

Figure 6-2 (Part 5 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SELDBR	Total number of logical data base reads by communications batch jobs.	PD (11,0)
SELDBW	Total number of logical data base writes by communications batch jobs.	PD (11,0)
SELDBU	Total number of miscellaneous data base operations by communications batch jobs.	PD (11,0)
SECMPT	Total number of communications writes by communications batch jobs.	PD (11,0)
SECMGT	Total number of communications reads by autostart jobs.	PD (11,0)
SEBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SEPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SACPU	Total processing unit time (in milliseconds) used by autostart jobs.	PD (11,0)
SARES1	Reserved.	PD (11,0)
SARES2	Reserved.	PD (11,0)
SAPRTL	Total number of print lines of all autostart jobs.	PD (11,0)
SAPRTP	Total number of print pages of all autostart jobs.	PD (11,0)
SASPD	Total time autostart jobs were suspended.	PD (11,0)
SARRT	Total time an autostart job waited during rerouting.	PD (11,0)
SANEW	Number of started autostart jobs.	PD (11,0)
SATERM	Number of ended autostart job.	PD (11,0)
SAPDBR	Total number of physical data base reads by autostart jobs.	PD (11,0)
SAPNDB	Total number of physical non-data-base reads by autostart jobs.	PD (11,0)
SAPWRT	Total number of physical writes by autostart jobs.	PD (11,0)
SALDBR	Total number of logical data base reads by autostart jobs.	PD (11,0)
SALDBW	Total number of logical data base writes by autostart jobs.	PD (11,0)
SALDBU	Total number of miscellaneous data base operations by autostart jobs.	PD (11,0)
SACMPT	Total number of communications writes by autostart jobs.	PD (11,0)
SACMGT	Total number of communications reads by autostart jobs.	PD (11,0)
SABRG	Number of PURGE(*YES) transactions.	PD (11,0)
SAPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SBCPU	Total processing unit time (in milliseconds) used by batch jobs.	PD (11,0)
SBRES1	Reserved.	PD (11,0)
SBRES2	Reserved.	PD (11,0)
SBPRTL	Total number of print lines of all batch jobs.	PD (11,0)
SBPRTP	Total number of print pages of all batch jobs.	PD (11,0)
SBSPD	Total time batch jobs were suspended.	PD (11,0)

Figure 6-2 (Part 6 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SBRRT	Total time a batch job waited during rerouting.	PD (11,0)
SBNEW	Number of started batch jobs.	PD (11,0)
SBTERM	Number of ended batch jobs.	PD (11,0)
SBPDBR	Total number of physical data base reads by batch jobs.	PD (11,0)
SBPNDB	Total number of physical non-data-base reads by batch jobs.	PD (11,0)
SBPWRT	Total number of physical writes by batch jobs.	PD (11,0)
SBLDBR	Total number of logical data base reads by batch jobs.	PD (11,0)
SBLDBW	Total number of logical data base writes by batch jobs.	PD (11,0)
SBLDBU	Total number of miscellaneous data base operations by batch jobs.	PD (11,0)
SBCMPT	Total number of communications writes by batch jobs.	PD (11,0)
SBCMGT	Total number of communications reads by batch jobs.	PD (11,0)
SBBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SBPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SICPU	Total processing unit time (in milliseconds) used by interactive jobs.	PD (11,0)
SITRNT	Total transaction time by interactive jobs.	PD (11,0)
SITRNS	Total number of transactions by interactive jobs.	PD (11,0)
SIPRTL	Total number of print lines of all interactive jobs.	PD (11,0)
SIPRTP	Total number of print pages of all interactive jobs.	PD (11,0)
SISPD	Total time interactive jobs were suspended.	PD (11,0)
SIRRT	Total time an interactive job waited during rerouting.	PD (11,0)
SINEW	Number of started interactive jobs.	PD (11,0)
SITERM	Number of ended interactive jobs.	PD (11,0)
SIPDBR	Total number of physical data base reads by interactive jobs.	PD (11,0)
SIPNDB	Total number of physical non-data-base reads by interactive jobs.	PD (11,0)
SIPWRT	Total number of physical writes by interactive jobs.	PD (11,0)
SILDBR	Total number of logical data base reads by interactive jobs.	PD (11,0)
SILDBW	Total number of logical data base writes by interactive jobs.	PD (11,0)
SILDBU	Total number of miscellaneous data base operations by interactive jobs.	PD (11,0)
SICMPT	Total number of communications writes by interactive jobs.	PD (11,0)
SICMGT	Total number of communications reads by interactive jobs.	PD (11,0)
SIBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SIPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SXCPU	Total processing unit time (in milliseconds) used by start CPF (SCPF) jobs/spool reader/spool writer.	PD (11,0)

Figure 6-2 (Part 7 of 8). System Performance Data (collected for each interval)

Field Name	Description	Attributes
SXRES1	Reserved.	PD (11,0)
SXRES2	Reserved.	PD (11,0)
SXPRTL	Total number of print lines of all SCPF jobs/spool reader/spool writer.	PD (11,0)
SXPRTP	Total number of print pages of all SCPF jobs/spool reader/spool writer.	PD (11,0)
SXSPD	Total time SCPF jobs/spool reader/spool writer were suspended.	PD (11,0)
SXRRT	Total time a SCPF jobs or spool reader/writer waited during rerouting.	PD (11,0)
SXNEW	Number of started SCPF jobs or spool reader/writer.	PD (11,0)
SXTERM	Number of ended SCPF jobs or spool reader/writer.	PD (11,0)
SXPDBR	Total number of physical data base reads by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXPNDB	Total number of physical non-data-base reads by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXPWRT	Total number of physical writes by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXLDBR	Total number of logical data base reads by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXLDBW	Total number of logical data base writes by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXLDBU	Total number of miscellaneous data base operations by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXCMPT	Total number of communications writes by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXCMGT	Total number of communications reads by SCPF jobs/spool reader/spool writer.	PD (11,0)
SXBRG	Number of PURGE(*YES) transactions.	PD (11,0)
SXPRG	Number of DYNAMIC PURGE(*NO) transactions.	PD (11,0)
SHCPU	Total processing unit time (in milliseconds) used by microcode/system jobs.	PD (11,0)
SMPLP	Machine pool paging: Number of pages transferred in and out of machine pool.	PD (11,0)
SMUPL	Highest user pool paging: Number of pages transferred in and out of user pool.	PD (11,0)
SUPL1	Pool with highest paging: Pool number with highest number of pages transferred in and out.	C (2)
SMXDU	Maximum disk utilization.	PD (11,0)
SMXDUI	Actuator with maximum utilization.	C (4)

Figure 6-2 (Part 8 of 8). System Performance Data (collected for each interval)

File Names: QAPMJOBS

JC/JI Data Base File Entries Figure 6-3 lists the various fields in the job completion (JC) and job interval (JI) file.

Note: In Figure 6-3, *job* means job/task.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) for job interval entry and job completion date, and time (hh:mm:ss) for job completion entry.	C (12)
INTSEC	Elapsed interval seconds.	PD (7,0)
JBSSYS	Name of the subsystem the job is running in.	C (10)
JBSLIB	Name of the library the subsystem is in.	C (10)
JBNAME	Job name/work station name.	C (10)
JBUSER	Job user.	C (10)
JBNBR	Job number.	C (6)
JBACCO	Job accounting code.	C (15)
JBTYPE	Job type (A,B,H,I,M,R,S,V,W,X) (see note 1).	C (1)
JBSTYP	Job subtype (see note 2).	C (1)
JBFLAG	Job flag (see note 3).	C (2)
JBS36E	Is job running in System/36 environment? (Y/N)	C (1)
	Reserved.	C (1)
JBPOOL	Job pool.	C (2)
JBPTY	Job priority.	C (3)
JBCPU	Processing unit time (in milliseconds) used.	PD (11,0)
JBRSP	Total transaction time (in seconds): Field has a value other than zero only if this is an interactive job.	PD (11,0)
JBSLC	Time-slice value (in milliseconds).	PD (11,0)
JB#TR	Number of transactions (5250 only): Field has a value other than zero only if this is an interactive job.	PD (11,0)
JBDBR	Number of data base reads: Total number of physical data base read operations for data base functions.	PD (11,0)
JBNDB	Number of non-data-base reads: Total number of physical non-data-base read operations for non-data-base functions.	PD (11,0)
JBWRT	Number of writes: Total number of physical write operations.	PD (11,0)
JBAW	Number of jobs going from active to wait: Total number of transitions from active state to wait state.	PD (11,0)
JBWI	Number of jobs going from wait to ineligible: Total number of transitions from wait state to ineligible state.	PD (11,0)

Figure 6-3 (Part 1 of 2). Job Data (one entry for each job in the system, collected for each interval)

Field Name	Description	Attributes
JBAI	Number of jobs going from active to ineligible: Total number of transitions from active state to ineligible state.	PD (11,0)
JBPLN	Number of print lines: Number of lines written by the program. This does not reflect what is actually printed. Spooled files can be ended, or printed with multiple copies.	PD (11,0)
JBPPG	Number of print pages.	PD (11,0)
JBPFL	Number of print files.	PD (11,0)
JBLWT	Number of data base writes (logical): Number of times the internal data base write function was called. This does not include I/O operations to readers/writers, or I/O operations caused by the CPYSPLF or DSPSPLF command. If SEQONLY(*YES) is specified, these numbers show each block of records read, not the number of individual records read.	PD (11,0)
JBLRD	Number of data base reads (logical): Number of times the data base module was called. This does not include I/O operations to readers/writers, or I/O operations caused by the CPYSPLF or DSPSPLF command. If SEQONLY(*YES) is specified, these numbers show each block of records read, not the number of individual records read.	PD (11,0)
JBDBU	Number of miscellaneous data base operations: Updates, deletes, force-end-of-data, and releases (logical).	PD (11,0)
JBCPT	Number of communications writes: These do not include remote work station activity. They include only activity related to OS/400ICF files when the I/O is for an OS/400-ICF device.	PD (11,0)
JBCGT	Number of communications reads (logical): These do not include remote work station activity. They include only activity related to OS/400-ICF files when the I/O is for an OS/400-ICF device.	PD (11,0)
JBSPD	Total suspended time (in milliseconds).	PD (11,0)
JBRRT	Total time job waited during reroutes (in milliseconds).	PD (11,0)
JBLND	Line description: Name of the communications line this work station and its controller is attached to. This is only available for remote work stations.	C (10)
JBCUD	Controller description: Name of the controller this work station is attached to.	C (10)
JB2LND	Secondary line description (pass-through and emulation only).	C (10)
JB2CUD	Secondary controller description (pass-through and emulation only).	C (10)
JBBRG	Number of times PAG was brought in: Total number of times the program access group (PAG) was brought into main storage after it was purged when running with PURGE(*YES) parameter specified.	PD (9,0)
JBPRG	Number of times PAG was purged: Total number of times the program access group (PAG) was purged when the job entered a long wait and was running with PURGE(*YES) parameter specified.	PD (9,0)
JBWSID	Work station address: The address of the work station on which the job is running. Byte 1 is the IOP bus number and address and bytes 2 and 3 are the device address.	C (3)

Figure 6-3 (Part 2 of 2). Job Data (one entry for each job in the system, collected for each interval)

Notes:

1. Job Types:
   - B = Batch
   - I = Interactive
   - A = Autostart
   - R = Spool reader
   - W = Spool writer
   - M = Subsystem monitor
   - S = System
   - X = SCPF job
   - H = HMC microcode
   - V = VMC microcode
2. Job Subtypes:
   - M = MRT (System/36 environment only)
   - E = Evoke (communications batch)
3. Job Flags:

Bit	
0	Pass-through source
1	Pass-through target
2	Emulation active
4	PC Support application
5	Target DDM job
6-15	Not used

File Name: QAPMDISK

DD File Entries In Figure 6-4, reference is made to an arm as opposed to a disk drive. The 9332 offers two configurations: 200M and 400M. The 200M version has one enclosure (one disk drive) with one actuator. The 400M version has one enclosure (one disk drive) with two actuators. The 9335 has one size: 850M with 2 actuators.: Figure 6-4 lists the fields in the disk data file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
DIOPID	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
DSUADR	Arm unit address: Byte 1 is the device port number for this arm; byte 2 is the arm number for that port. For 9332, byte 2 can be 0 or 1; for the 9335, 0 to 7; for the 6100, 0.	C (2)
DSARM	Disk arm number: Specifies the unique identifier of the unit. Each actuator arm on the disk drives available to the machine represents a unit of auxiliary storage. The value of the unit number is assigned by the system when the unit is allocated to an ASP.	C (4)
DSTYPE	Disk drive type such as 9332, 9335, or 6100.	C (4)
DSRDS	Number of Read Data commands.	PD (5,0)
DSWRTS	Number of the Write Data commands.	PD (5,0)
DSSCAN	Number of Search String commands: This count is always zero, because Search String commands are not supported for the 9332, 9335, or 6100.	PD (5,0)
DSBLKR	Number of blocks read: The block length is 520 bytes, which includes 8 bytes of system control information.	PD (11,0)
DSBLKW	Number of blocks written: The block length is 520 bytes, which includes 8 bytes of system control information.	PD (11,0)
DSBUFO	Number of buffer overruns: The number of times that data was available to be read into the disk controller buffer from the disk, but the disk controller buffer still contained valid data that was not retrieved by the storage device controller. Consequently, the disk had to take an additional revolution until the buffer was available to accept data. This field is 0 for disk drive type 6100.	PD (5,0)
DSBUFU	Number of buffer underruns: The number of times that the disk controller was ready to transfer data to the disk on a write, but the disk controller buffer was empty. The data was not transferred in time by the disk IOP to the disk controller buffer. The disk was forced to take an extra revolution awaiting the data. This field is 0 for disk drive type 6100.	PD (5,0)

Figure 6-4 (Part 1 of 3). Disk Storage Data (one for each actuator)

Field Name	Description	Attributes
DSIDLC	Processor idle loop counter (see note 1): The number of times the disk controller passed through the idle loop. This count is increased differently for the 9332 and the 9335. For the 9332, this counter is increased only if the disk controller is totally idle (for example, no I/O operations are active). For the 9335, even though the disk controller may be idle and the counter gets increased, an I/O operation can be active (for example, seek is being performed). This field is 0 for disk drive type 6100.	PD (11,0)
DSIDLT	Processor idle loop time (see note 4): The time (in hundredths of microseconds) to make one pass through the idle loop. This field is 0 for disk drive type 6100.	PD (11,0)
DSSK1	Number of seeks > 2/3 (see note 2): The number of times the arm traveled more than 2/3 of the disk on a seek.	PD (11,0)
DSSK2	Number of seeks > 1/3 and < 2/3 (see note 2): The number of times the arm traveled more than 1/3 but less than 2/3 of the disk on a seek.	PD (11,0)
DSSK3	Number of seeks > 1/6 and < 1/3 (see note 2): The number of times the arm traveled more than 1/6 but less than 1/3 of the disk on a seek.	PD (11,0)
DSSK4	Number of seeks > 1/12 and < 1/6 (see note 2): The number of times the arm traveled more than 1/12 but less than 1/6 of the disk on a seek.	PD (11,0)
DSSK5	Number of seeks < 1/12 (see note 2): The number of times the arm traveled from its current position but less than 1/12 of the disk on a seek.	PD (11,0)
DSSK6	Number of zero seeks (see note 1): The number of times the access arm did not physically move on a seek request. The operation may have resulted in a head switch. This field is 0 for disk drive type 6100. The number of zero seeks will be accumulated in DSSK5.	PD (11,0)
DSQUEL	Average queue length (see note 5): The number of I/O operations waiting service at sample time. This number includes the I/O operation that is in progress.	PD (11,0)
DSNBSY	Number of times arm not busy (see note 5): The number of times there were no outstanding I/O operations active at sample time.	PD (11,0)
DSSMPL	Number of samples taken at two per second (see note 5): The number of samples taken at approximately two per second for the DSQUEL and DSNBSY fields.	PD (11,0)
DSCAP	Drive capacity (in bytes): Total number of bytes of auxiliary storage provided on the unit for the storage of objects and internal machine functions when the ASP containing it is not under check sum protection. The unit reserved system space value is subtracted from the unit capacity to calculate this capacity. When the ASP containing the unit is under check sum protection, the unit check sum information describes how much of the unit capacity is used for protected space, unprotected space, and redundant data.	PD (11,0)
DSAVL	Drive available space (in bytes): Total number of bytes of auxiliary storage space that is not currently assigned to objects or internal machine functions, and therefore is available on the unit if the ASP containing it is not under check sum protection.	PD (11,0)

Figure 6-4 (Part 2 of 3). Disk Storage Data (one for each actuator)

Field Name	Description	Attributes
DSASP	ASP number: Specifies the ASP to which this unit is currently allocated. A value of 1 specifies the system ASP. A value from 2 through 16 specifies a user ASP. A value of 0 indicates that this unit is currently not allocated.	C (2)
DSCSS	Check sum number: Specifies the check sum set to which this unit is currently allocated. A value from 1 through 16 specifies a check sum set. A value of 0 specifies that the unit is currently not assigned to a check sum set.	C (2)
DSPCAP	Permanent storage capacity: Number of bytes of auxiliary storage formatted for storage of protected data on the unit. This field has a value other than zero if this unit is allocated to a check sum set. Units not allocated to a check sum set contain no permanent storage area. This value does not include the size of any data redundancy area which may also be formatted on the unit.	PD (11,0)
DSPAVL	Permanent storage space available: Number of bytes of permanent space on auxiliary storage available for allocation on the unit that is not currently assigned to objects of internal machine functions. This field has a value other than zero only if this unit is allocated to a check sum set. Units not allocated to a check sum set contain no protected storage area.	PD (11,0)

Figure 6-4 (Part 3 of 3). Disk Storage Data (one for each actuator)

Notes:

1. 9332/9335 inconsistencies:
   - 9335 updates the idle count only when the processing unit (A) is not busy. Disk operations such as seek could be in progress. 9332 updates the idle count when there is no activity in any of its processors.
   - If there is no movement and no head switch, the 9332 does not count this operation as a zero seek, the 9335 does.
   - If an operation causes a head switch (starts a read or write on one track and end up on another track), the 9332 counts this as a zero seek, the 9335 does not.
2.

9335:	9332:
> 2/3	> = 2/3
> 1/3 and < = 2/3	> = 1/3 and < 2/3
> 1/6 and < = 1/3	> = 1/6 and < 1/3
> 1/12 and < = 1/6	> = 1/12 and < 1/6
< = 1/12	< 1/12

3. DSIDLC and DSIDLT are duplicated across all arms for the same disk controller.
4. The idle loop count and time are used to calculate the storage device controller utilization as follows:

   From the interval time, subtract the product of the idle loop count times the idle loop time. Divide the result by the interval time. For example:

   ```
   IOP utilization = (INTSEC - (DSIDLC x DSIDLT))/ INTSEC
   ```

5. The following three counts are extracted from the QAPMDIOP record:
   - DSSMPL
   - DSQUEL
   - DSNBSY

6. The following formulas describe how several of the fields in the previous table can be used to calculate utilization, response time, service time, and wait time for each arm.

   Arm utilization (DSUTL): That part of the total interval expressed as a percentage that the arm was being used for I/O operations.

   ```
   DSUTL = Arm Busy - (DSSMPL - DSNBSY)/DSSMPL
   ```

   Arm accesses per second (DSAS): The number of reads and writes for this arm during the interval.

   ```
   DSAS = DSRDS + DSWRTS/INTSEC
   ```

   Service time (DSSRVCT): The average time the arm was busy performing I/O operations.

   ```
   DSSRVCT = DSUTL/DSAS
   ```

File Name: QAPMPOOL

Main Storage Pool File Entries Figure 6-5 lists the fields in the storage pool file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
PONBR	Pool number: Specifies the unique identifier of this pool. The value is from 1 to 16.	C (2)
POACTL	Pool activity level setting: The maximum number of processes that can be active in the machine at the same time.	PD (3,0)
POSIZ	Pool size (in kilobytes): The amount of main storage assigned to the pool.	PD (7,0)
PORES	Pool reserved size (in kilobytes): Specifies the amount of storage from the pool that is dedicated to machine functions.	PD (5,0)
PODBF	Pool data base faults: Total number of interruptions to processes (not necessarily assigned to this pool) that were required to transfer data into the pool to permit the MI instruction to process the data base function.	PD (11,0)
PONDBF	Pool non-data-base faults: Total number of interruptions to processes (not necessarily assigned to this pool) that were required to transfer data into the pool to permit the MI instruction to process non-data-base functions.	PD (11,0)
PODBPG	Pool data base pages read/written: Total number of pages of data base data transferred from auxiliary storage to the pool to permit the instruction to run as a consequence of set access state, implicit access group movement, and internal machine actions.	PD (11,0)
PONDPG	Pool non-data-base pages read/written: Total number of pages of data base data transferred from auxiliary storage to the pool to permit the instruction to run as a consequence of set access state, implicit access group movement, and internal machine actions.	PD (11,0)
POAW	Number of active to wait transitions: Total number of transitions by processes assigned to this pool from active state to wait state.	PD (11,0)
POWI	Number of wait to ineligible: Total number of transitions by processes assigned to this pool from wait state to ineligible state.	PD (11,0)
POAI	Number of active to ineligible: Total number of transitions by processes assigned to this pool from active state to ineligible state.	PD (11,0)

Figure 6-5. Main Storage Data (one for each storage pool)

File Name: QAPMHDLC

HDLC File Entries In Figure 6-6, statistics are kept on a line basis for the fields in the high-level data link control (HDLC) file.

Field Name	Description	Attributes
INTNUM	Interval number: the nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
SHIOP	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
SHLND	Line description: The name of the description for this line.	C (10)
	Reserved.	C (10)
SHLSP	Line speed: The speed of the line in bits per second bps.	PD (11,0)
SHBTRN	Bytes transmitted: The number of bytes transmitted including bytes transmitted again.	PD (11,0)
SHBRCV	Bytes received: The number of bytes received including all bytes in frames that had any kind of error.	PD (11,0)
SHPRCL	Protocol type: S for SDLC.	C (1)
SHFTRN	Number of frames transmitted (I, supervisory, and frames not numbered) excluding frames transmitted again.	PD (11,0)
SHIFTR	Number of I-frames transmitted excluding I-frames transmitted again.	PD (11,0)
SHIFRT	Number of I-frames transmitted again.	PD (11,0)
SHFRT	Number of I, supervisory, and frames not numbered transmitted again.	PD (11,0)
SHEFFR	Error-free frames received: The number of I, supervisory, and frames not numbered received without error (whether or not they were transmitted again from the remote side).	PD (11,0)
SHEFIR	Error-free I-frames received: The number of I-frames received without error (whether or not they were transmitted again from the remote side).	PD (11,0)
SHFRIE	Frames received in error: The number of I, supervisory, and frames not numbered received in error. There are three error possibilities: (1) a supervisory or I-frame was received with an Nr count that is requesting retransmission of a frame, (2) an I-frame was received with an Ns count that indicates that frames were missed, (3) a frame is received with one of the following errors – a frame check sequence error, an abnormal end, a receive overrun, or a frame truncated error.	PD (11,0)

Figure 6-6 (Part 1 of 2). SDLC/HDLC Statistics

Field Name	Description	Attributes
SHIFR	Invalid frames received: The number of invalid frames received. These are frames received with either: (1) short frame error – frame is less than 32 bits or (2) residue error – frame is not on a byte boundary.	PD (11,0)
SHRRFT	Number of receive ready supervisory frames transmitted.	PD (11,0)
SHRRFR	Number of receive ready supervisory frames received.	PD (11,0)
SHRNRT	Number of receive not ready supervisory frames transmitted.	PD (11,0)
SHRNRR	Number of receive not ready supervisory frames received.	PD (11,0)
SHLNKR	Data link resets: The number of times a set normal response mode (SNRM) was received when the station was already in normal response mode.	PD (11,0)

Figure 6-6 (Part 2 of 2). SDLC/HDLC Statistics

File Name: QAPMASYN

Asynchronous File Entries Figure 6-7 lists the fields in the asynchronous file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
AIOPID	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
ASLND	Line description: The name of the description for this line.	C (10)
	Reserved.	C (10)
ASLSP	Line speed: The speed of this line in bits per second (bps)	PD (11,0)
ASBTRN	Number of bytes transmitted (data and control characters) including bytes transmitted again because of errors.	PD (11,0)
ASBRCV	Number of bytes received (data and control characters), including characters received in error.	PD (11,0)
ASPRCL	Protocol type: A for asynchronous.	C (1)
ASPDUR	The total number of protocol data units received.	PD (11,0)
ASPDUE	The total number of protocol data units received with parity and stop bit errors.	PD (11,0)
ASPDUT	The total number of protocol data units successfully transmitted and the data-circuit terminating equipment (DCE) acknowledged.	PD (11,0)

Figure 6-7. Asynchronous Statistics (one per line)

File Name: QAPMBSC

Binary Synchronous File Entries Figure 6-8 lists the fields in the binary synchronous file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
BIOPID	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
BSLND	Line description: The name of the description for this line.	C (10)
	Reserved.	C (10)
BSLSP	Line speed: The speed of the line in bits per second (bps).	PD (11,0)
BSBTRN	Bytes transmitted: The number of bytes (data and control characters) transmitted, including bytes transmitted again.	PD (11,0)
BSBRCV	Bytes received: The number of bytes (data and control characters) received including bytes received in error.	PD (11,0)
BSPRCL	Protocol type: B for binary synchronous.	C (1)
BSDCRV	Data characters received: The number of data characters received successfully (excluding synchronous characters) while in data mode. For feature types 2507 and 6150, this value is equal to field BSBRCV.	PD (11,0)
BSDCRE	Data characters received in error: The number of data characters received with a block-check character error while in data mode. For feature types 2507 and 6150, this value is equal to field BSCRER.	PD (11,0)
BSDCTR	Data characters transmitted: The number of data characters transmitted successfully while in data mode. For feature types 2507 and 6150, this value is equal to field BSBTRN.	PD (11,0)
BSCRER	Characters received in error: The number of characters received with a block-check character error.	PD (11,0)
BSLNK	Negative acknowledgment character received to text sent (see note): The number of times the remote station or device did not understand the command sent from the host system.	PD (11,0)
BSLWA	Wrong acknowledgment character to text sent (see note): The host system received an acknowledgment from the remote device that was not expected. For example, the system expected an ACK0 and received an ACK1.	PD (11,0)

Figure 6-8 (Part 1 of 3). BSC Statistics (one per link/station)

Field Name	Description	Attributes
BSLQTS	Enqueue to text sent (see note): Text was sent by a station and an ENQ character was returned. The receiving station expected some form of acknowledgment, such as an ACK0, ACK1, or NAK.	PD (11,0)
BSLINV	Invalid (unrecognized format): One of the delimiter characters that encloses the data in brackets being sent/received is invalid (see note).	PD (11,0)
BSLQAK	Enqueue to acknowledged character: The remote station returned an acknowledgment (for example, ACK0) and the host system sent an ENQ character. This indicates that the host station did not recognize the acknowledgment as a valid acknowledgment (see note).	PD (11,0)
BSLTNK	Negative acknowledgment character received to text sent (total): The number of times the remote station did not understand the command sent from the host system (see note).	PD (11,0)
BSLTWA	Wrong acknowledgment character to text sent (total): The host system received an acknowledgment from the remote device that was not expected. For example, the host system expected an ACK0 and received an ACK1 (see note).	PD (11,0)
BSLTQT	Enqueue to text sent (total): Text was sent by a station and an ENQ character was returned. The receiving station expected some form of acknowledgment such as an ACK0, ACK1, or NAK (see note).	PD (11,0)
BSLTIV	Invalid (unrecognized format) (total): One of the delimiter characters that enclose the data in brackets being sent/received is invalid (see note).	PD (11,0)
BSLTQA	Enqueue to acknowledged character (total): The remote station returned an acknowledgment (for example, ACK0) and the host station sent an ENQ character. This indicates that the host station did not recognize the acknowledgment as a valid acknowledgment (see note).	PD (11,0)
BSLDRA	Disconnect received: The remote station issued a disconnect with abnormal end. This could occur when error recovery did not succeed or the binary synchronous job was ended.	PD (11,0)
BSLEAB	End of transmission (EOT) received (abnormal end): Similar to a disconnect.	PD (11,0)
BSLDFA	Disconnect received (forward abnormal end): The host station issued a disconnect with abnormal end. This could occur when the error recovery did not succeed, or the binary synchronous job was ended.	PD (11,0)
BSLEFA	EOT received (forward abnormal end): Similar to a disconnect.	PD (11,0)
BSLDBT	Number of data blocks transmitted.	PD (11,0)
BSLDBR	Number of data blocks received.	PD (11,0)
BSLBKR	Number of data blocks transmitted again.	PD (11,0)
BSLBKE	Number of data blocks received in error.	PD (11,0)
BSLTRT	Total number of characters transmitted again, including control characters.	PD (11,0)
BSLDRT	Total number of data characters transmitted again.	PD (11,0)

Figure 6-8 (Part 2 of 3). BSC Statistics (one per link/station)

Field Name	Description	Attributes

Note: The counters BSLNK through BSLQAK are error recovery counters, and are increased the first time the error is detected. The counters BSLTNK to BSLTQA are error recovery counters, and are increased every time the error occurs. The same errors are being counted in each set of counters, so the first set indicates how many times an error was detected, and the second set indicates how many retries it took to recover from the errors.

Figure 6-8 (Part 3 of 3). BSC Statistics (one per link/station)

File Name: QAPMX25

X.25 File Entries Figure 6-9 lists the fields in the X.25 file.: In Figure 6-9, the XH prefix in the label refers to HDLC counters, XL refers to X.25 Logical Link Control (LLC) counters, and XP refers to Packet Level Control (PLC) counters.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
XIOPID	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
XLLND	Line description: The name of the description for this line.	C (10)
	Reserved.	C (10)
XLLSP	Line speed: The speed of this line in bits per second (bps).	PD (11,0)
XHBTRN	Bytes transmitted: The number of bytes transmitted, including bytes transmitted again.	PD (11,0)
XHBRCV	Bytes received: The number of bytes received, including all bytes in frames that had any kind of error.	PD (11,0)
XHPRCL	Protocol type: X for X.25.	C (1)
XHFTRN	Frames transmitted: The number of frames transmitted (I, supervisory, and frames not numbered), excluding frames transmitted again.	PD (11,0)
XHIFTR	I-frames transmitted: The number of I-frames transmitted, excluding I-frames transmitted again.	PD (11,0)
XHIFRT	I-frames transmitted again: The number of I-frames transmitted again.	PD (11,0)
XHFRT	Frames transmitted again: The number of I, supervisory, and frames not numbered transmitted again.	PD (11,0)
XHEFFR	Error-free frames received: The number of I, supervisory, and frames not numbered received without error (whether or not they were transmitted again from the remote side).	PD (11,0)
XHEFIR	Error-free I-frames received: The number of I-frames received without error (whether or not they were transmitted again from the remote side).	PD (11,0)

Figure 6-9 (Part 1 of 3). X.25 Statistics

Field Name	Description	Attributes
XHFRIE	Frames received in error: The number of I, supervisory, and frames not numbered received in error. There are three error possibilities: (1) a supervisory or I-frame was received with an Nr count that is requesting retransmission of a frame, (2) an I-frame was received with an Ns count that indicates that frames were missed, (3) a frame was received with one of the following errors – a frame check sequence error, an abnormal end, a receive overrun or a frame truncated error.	PD (11,0)
XHIFR	Invalid frames received: The number of invalid frames received. These are frames received with either: (1) a short frame error – frame is less than 32 bits, or (2) a residue error – frame is not on a byte boundary.	PD (11,0)
XHRRFT	Number of receive ready supervisory frames transmitted.	PD (11,0)
XHRRFR	Number of receive ready supervisory frames received.	PD (11,0)
XHRNRT	Number of receive not ready supervisory frames transmitted.	PD (11,0)
XHRNRR	Number of receive not ready supervisory frames received.	PD (11,0)
XHLNKR	Link resets: The number of times when a set normal response mode (SNRM) was received when the station was already in normal response mode.	PD (11,0)
XLLCID	Logical channel id currently in use.	C (2)
XLLSS	State of the link station: 0—Not active 1—Opened station 2—Awaiting reconnection 3—Reconnecting 4—Closed station	C (1)
XLLLPS	Logical link protocol state: 0—Disconnected 1—Connecting 2—Connected 3—Disconnecting	C (1)
XLITR	Interface protocol data units transmitted (LLC level).	PD (11,0)
XLIRC	Interface protocol data units received.	PD (11,0)
XLIRT	Interface protocol data units transmitted again.	PD (11,0)
XLIRE	Interface protocol data units received in error (check sum).	PD (11,0)
XLLXTR	Number of XIDS transmitted.	PD (11,0)
XLXRC	Number of XIDS received.	PD (11,0)
XLTT	Number of tests transmitted.	PD (11,0)
XLTR	Number of tests received.	PD (11,0)
XLLJT	Number of LLC rejects transmitted.	PD (11,0)
XLLJR	Number of LLC rejects received.	PD (11,0)
XLRLD	Number of received LLC protocol data units discarded.	PD (11,0)

Figure 6-9 (Part 2 of 3). X.25 Statistics

Field Name	Description	Attributes
XLTO	Number of time-outs.	PD (11,0)
XLCED	Check sum errors detected.	PD (11,0)
XLSRA	Successful recovery attempts.	PD (11,0)
XLRA	Recovery attempts.	PD (11,0)
XLRSI	Number of reset indications from packet-link control.	PD (11,0)
XLCLS	Number of close station indications from packet-link control.	PD (11,0)
XLRNR	LLC receive not ready frames received.	PD (11,0)
XPLCID	Logical channel ID.	C (2)
XPLSS	State of the logical channel. Switched Virtual Circuit: 0—Logical channel not connected 1—Logical channel awaiting connection—out 2—Logical channel awaiting connection—in 3—Logical channel connected 4—Logical channel awaiting disconnection Permanent Virtual Circuit: 0—Logical channel not opened 3—Logical channel in data transfer state 4—Logical channel being reset	C (1)
XPTPT	Total packets transmitted.	PD (11,0)
XPTPR	Total packets received.	PD (11,0)
XPDPT	Data packets transmitted.	PD (11,0)
XPDPR	Data packets received.	PD (11,0)
XPRPT	Reset packets transmitted.	PD (11,0)
XPROR	Reset packets received.	PD (11,0)
XPRRT	Restart requests transmitted.	PD (11,0)
XPRIR	Restart indications received.	PD (11,0)
XPRNR	Receive not ready packets received.	PD (11,0)

Figure 6-9 (Part 3 of 3). X.25 Statistics

File Name: QAPMECL

Token-Ring Network File Entries Figure 6-10 lists the fields in the token-ring local area network file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
EIOPI	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
ELLND	Line description: The name of the description for this line.	C (10)
	Reserved.	C (10)
ELLSP	Line speed: The line speed expressed in bits per second (bps).	PD (11,0)
ELTFT	Total number of Type II frames transmitted.	PD (11,0)
ELTFR	Total number of Type II frames received.	PD (11,0)
ELIFT	Total number of I-frames transmitted.	PD (11,0)
ELIFR	Total number of I-frames received.	PD (11,0)
ELICT	Total number of characters transmitted in all I-frames.	PD (11,0)
ELICR	Total number number of characters received in all I-frames.	PD (11,0)
ELPRCL	Protocol type: E for token-ring network.	C (1)
ELRFT	Number of receive not ready frames transmitted.	PD (5,0)
ELRFR	Number of receive not ready frames received.	PD (5,0)
ELFFT	Number of frame reject frames transmitted.	PD (5,0)
ELFFR	Number of frame reject frames received.	PD (5,0)
ELRJFR	Number of reject frames received.	PD (5,0)
ELRJFT	Number of reject frames transmitted.	PD (5,0)
ELSFT	Number of set asynchronous balanced mode extended frames transmitted.	PD (5,0)
ELSFR	Number of set asynchronous balanced mode extended frames received.	PD (5,0)
ELDFT	Number of disconnect frames transmitted.	PD (5,0)
ELDFR	Number of disconnect frames received.	PD (5,0)
ELDMT	Number of disconnect mode frames transmitted.	PD (5,0)

Figure 6-10 (Part 1 of 3). Token-Ring Network Counter

Field Name	Description	Attributes
ELDMR	Number of disconnect mode frames received.	PD (5,0)
ELN2R	N2 retries end count: This count is updated when the host has attempted to contact a station n times and n times the T1 timer ended before the station responded.	PD (5,0)
ELTIT	T1 timer end count: Number of times the T1 timer ended. See ELN2R description.	PD (5,0)
EMFTR	Total frames transmitted: Total number of frames (LLC and MAC) transmitted.	PD (5,0)
EMFRV	Total frames received: Total number of frames (LLC and MAC) received.	PD (5,0)
EMMFT	MAC frames transmitted: Total number of MAC frames transmitted.	PD (5,0)
EMMFR	MAC frames received: Total number of MAC frames received.	PD (5,0)
EMRIT	Routing information frames transmitted: Total number of frames (LLC and MAC) with a routing-information field transmitted.	PD (5,0)
EMRIR	Routing information frames received: Total number of frames (LLC and MAC) with a routing-information field received.	PD (5,0)
EMLNE	Line error: Code violation of frame-check sequence error.	PD (5,0)
EMINE	Internal error: Adapter internal error.	PD (5,0)
EMBRE	Burst error: Burst of same polarity is detected by the physical unit after the starting delimiter of a frame or token.	PD (5,0)
EMAFE	Address-recognized indicator or frame-copied indicator error: Physical control field-extension field error.	PD (5,0)
EMABT	Abnormal ending delimiter: Abnormal ending delimiter transmitted because of internal error.	PD (5,0)
EMLST	Lost frame: Physical trailer timer ended while IOA is in transmit stripping state.	PD (5,0)
EMRXC	Receive congestion: Frame not copied because no buffer was available for the IOA to receive.	PD (5,0)
EMFCE	Frame-copied error: The frame with a specific destination address was copied by another adapter.	PD (5,0)
EMFQE	Frequency error on the adapter.	PD (5,0)
EMTKE	Token error: The adapter any token timer ended without detecting any frame or token.	PD (5,0)
EMDBE	Direct memory access bus error: IOP/IOA bus DMA error.	PD (5,0)
EMDPE	Direct memory access parity error: IOP/IOA DMA parity error.	PD (5,0)
EMANR	Total number of frames with address not recognized error.	PD (5,0)
EMFNC	Total number of frames with frame not copied error.	PD (5,0)
EMTSE	Total number of adapter frame transmit or frame strip process errors.	PD (5,0)

Figure 6-10 (Part 2 of 3). Token-Ring Network Counter

Field Name	Description	Attributes
EMUAP	Unauthorized access priority: The access priority requested is not authorized.	PD (5,0)
EMUMF	Unauthorized MAC frame: The adapter is not authorized to send a MAC frame with the source class specified, or the MAC frame has a source class of zero, or the MAC frame physical control field attention field is > 1.	PD (5,0)
EMSFT	Soft error: Total number of soft errors as reported by the adapter.	PD (5,0)
EMTBC	Total number of beacon frames transmitted.	PD (5,0)
EMIOA	IOA status overrun: Adapter interrupt status queue overrun, earliest status discarded.	PD (5,0)
EMFDC	Total number of frames discarded.	PD (5,0)
EMSIN	Total number of interrupts that MAC could not decode.	PD (11,0)

Figure 6-10 (Part 3 of 3). Token-Ring Network Counter

File Name: QAPMCIOP

Communications Controller File Entries Figure 6-11 lists the fields in the communications controller file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
CIIOP	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
CTIPKT	Total packets transferred.	PD (11,0)
CIDMAO	Total bytes transmitted from an IOP to the system across the bus.	PD (11,0)
CIDMAI	Total bytes transmitted to the IOP from the system across the bus.	PD (11,0)
CIOPSR	OPSTART bus unit message received from another bus unit using normal flow.	PD (11,0)
CIOPSS	OPSTART bus unit message received from another bus unit using reverse flow method 2 (always 0).	PD (11,0)
CISGLR	Signals received.	PD (11,0)
CIOPST	OPSTARTS sent.	PD (11,0)
CISLGS	Signals sent.	PD (11,0)
CIRSTQ	Restart queues sent.	PD (11,0)
CIRQDO	DMA REQUESTS sent for output of data.	PD (11,0)
CIRQDI	Direct storage device requests sent for input of data.	PD (11,0)
CIBNAR	Occurrences of BNA received.	PD (11,0)
CIPRCU	Processor utilization: The number of fixed-time intervals that this communications IOP spent in the idle state.	PD (11,0)
CIIDLC	Idle loop count (see note): The number of times the communications IOP ran an idle loop. This is done when the IOP has no work to perform. This count is used with the idle loop time.	PD (11,0)
CIIDLT	Idle loop time (see note): The time (in hundredths of microseconds) to run the idle loop once.	PD (11,0)
CIRAMU	Available direct-access storage (in bytes): The number of bytes of free direct-access storage in the IOP. The free direct-access storage will probably be non-contiguous because of fragmentation.	PD (11,0)

Figure 6-11. Communications Controller IOP Data (one for each communications controller)

Note: The idle loop count and time are used to calculate the communications IOP utilization as follows:

From the interval time, subtract the product of the idle loop count times the idle loop time. Divide the result by the interval time. For example:

```
IOP utilization = (INTSEC - (CIIDLC * CIIDLT))/ INTSEC
```

File Name: QAPMDIOP

Storage Device Controller File Entries Figure 6-12 lists the fields in the storage device controller file.

Note: In Figure 6-12, *device* means disk.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
DIIOP	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
DIIDLC	Idle loop count (see note): The number of times the disk controller IOP ran an idle loop. This is done when the IOP has no work to perform. This count is used with the idle loop time.	PD (11,0)
DIIDLT:	Idle loop time (see note): The time (in hundredths of microseconds) to run the idle loop once.	PD (11,0)
DIRID0	Device 0 resource name: Device resource name used to refer to this device. The storage devices are attached on these ports, and there are 16 ports.	C (8)
DISMP0	Number of sample of device 0: The number of times during the data collection interval that the device 0 queue was sampled to see if any I/O requests were waiting for service.	PD (11,0)
DIQLN0	Device 0 queue length: The number of I/O requests waiting for service on the device 0 queue at sample time.	PD (11,0)
DINRQ0	No-request-serviced count: The number of times that no I/O requests were waiting for service when the queue was sampled.	PD (11,0)
DIRID1	Device 1 resource name.	C (8)
DISMP1	Number of samples on device 1.	PD (11,0)
DIQLN1	Device 1 queue length.	PD (11,0)
DINRQ1	No-request-serviced count.	PD (11,0)
DIRID2	Device 2 resource name.	C (8)
DISMP2	Number of samples on device 2.	PD (11,0)
DIQLN2	Device 2 queue length.	PD (11,0)
DINRQ2	No-request-serviced count.	PD (11,0)
DIRID3	Device 3 resource name.	C (8)

Figure 6-12 (Part 1 of 3). Storage Device Controller IOP Data (one for each controller)

Field Name	Description	Attributes
DISMP3	Number of samples on device 3.	PD (11,0)
DIQLN3	Device 3 queue length.	PD (11,0)
DINRQ3	No-request-serviced count.	PD (11,0)
DIRID4	Device 4 resource name.	C (8)
DISMP4	Number of samples on device 4.	PD (11,0)
DIQLN4	Device 4 queue length.	PD (11,0)
DINRQ4	No-request-serviced count.	PD (11,0)
DIRID5	Device 5 resource name.	C (8)
DISMP5	Number of samples on device 5.	PD (11,0)
DIQLN5	Device 5 queue length.	PD (11,0)
DINRQ5	No-request-serviced count.	PD (11,0)
DIRID6	Device 6 resource name.	C (8)
DISMP6	Number of samples on device 6.	PD (11,0)
DIQLN6	Device 6 queue length.	PD (11,0)
DINRQ6	No-request-serviced count.	PD (11,0)
DIRID7	Device 7 resource name.	C (8)
DISMP7	Number of samples on device 7.	PD (11,0)
DIQLN7	Device 7 queue length.	PD (11,0)
DINRQ7	No-request-serviced count.	PD (11,0)
DIRID8	Device 8 resource name.	C (8)
DISMP8	Number of samples on device 8.	PD (11,0)
DIQLN8	Device 8 queue length.	PD (11,0)
DINRQ8	No-request-serviced count.	PD (11,0)
DIRID9	Device 9 resource name.	C (8)
DISMP9	Number of samples on device 9.	PD (11,0)
DIQLN9	Device 9 queue length.	PD (11,0)
DINRQ9	No-request-serviced count.	PD (11,0)
DIRIDA	Device 10 resource name.	C (8)
DISMPA	Number of samples on device 10.	PD (11,0)
DIQLNA	Device 10 queue length.	PD (11,0)
DINRQA	No-request-serviced count.	PD (11,0)
DIRIDB	Device 11 resource name.	C (8)
DISMPB	Number of samples on device 11.	PD (11,0)

Figure 6-12 (Part 2 of 3). Storage Device Controller IOP Data (one for each controller)

Field Name	Description	Attributes
DIQLNB	Device 11 queue length.	PD (11,0)
DINRQB	No-request-serviced count.	PD (11,0)
DIRIDC	Device 12 resource name.	C (8)
DISMPC	Number of samples on device 12.	PD (11,0)
DIQLNC	Device 12 queue length.	PD (11,0)
DINRQC	No-request-serviced count.	PD (11,0)
DIRIDD	Device 13 resource name.	C (8)
DISMPD	Number of samples on device 13.	PD (11,0)
DIQLND	Device 13 queue length.	PD (11,0)
DINRQD	No-request-serviced count.	PD (11,0)
DIRIDE	Device 14 resource name.	C (8)
DISMPE	Number of samples on device 14.	PD (11,0)
DIQLNE	Device 14 queue length.	PD (11,0)
DINRQE	No-request-serviced count.	PD (11,0)
DIRIDF	Device 15 resource name.	C (8)
DISMPF	Number of samples on device 15.	PD (11,0)
DIQLNF	Device 15 queue length.	PD (11,0)
DINRQF	No-request-serviced count.	PD (11,0)
DITPDK	Total packets transferred.	PD (11,0)
DIDMAO	Total bytes transmitted from the IOP to the system across the bus.	PD (11,0)
DIDMAI	Total bytes transmitted to the IOP from the system across the bus.	PD (11,0)
DIOPSR	OPSTART bus unit message received from another bus unit using normal flow.	PD (11,0)
DIOPSS	OPSTART bus unit message received from another bus unit using reverse flow method 2 (always 0).	PD (11,0)
DISGLR	Signals received.	PD (11,0)
DIOPST	OPSTARTS sent.	PD (11,0)
DISGLS	Signals sent.	PD (11,0)
DIRSTQ	Restart queues sent.	PD (11,0)
DIRQDO	Direct storage device requests sent for output of data.	PD (11,0)
DIRQDI	Direct storage device requests sent for input of data.	PD (11,0)
DIBNAR	Occurrences of BNA received.	PD (11,0)

Figure 6-12 (Part 3 of 3). Storage Device Controller IOP Data (one for each controller)

File Name: QAPMLIOP

Twinaxial IOP Data File Entries Figure 6-13 lists the fields in the twinaxial IOP data file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
LIIOP	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
LIRIDC	Resource ID of controller: The resource ID of the IOP.	C (8)
LITPKT	Total packets transferred.	PD (11,0)
LIDMAO	Total bytes transmitted from the IOP to the system across the bus.	PD (11,0)
LIDMAI	Total bytes transmitted to the IOP from the system across the bus.	PD (11,0)
LIOPSR	OPSTART bus unit message received from another bus unit using normal flow.	PD (11,0)
LIOPSS	OPSTART bus unit message received from another bus unit using reverse flow method 2.	PD (11,0)
LISGLR	Signal bus unit message received from another bus unit.	PD (11,0)
LIOPST	OPSTARTS sent to another bus unit using reverse flow method 2.	PD (11,0)
LISGLS	Signals sent to another bus unit.	PD (11,0)
LIRSTQ	Restart queues bus unit message sent to another bus unit.	PD (11,0)
LIRQDO	Direct storage device requests sent for output of data.	PD (11,0)
LIRQDI	Direct storage device requests sent for input of data.	PD (11,0)
LIBNAR	Occurrences of BNA received.	PD (11,0)
LIIOQC	Wait-on-I/O queue count: The number of I/O requests on the wait-on-I/O queue at sample time. The wait-on-I/O queue holds I/O requests that are being processed or waiting to be processed.	PD (11,0)
LISQC	Suspend queue count: The number of elements on the suspend queue at sample time.	PD (11,0)
LIAQC	Active queue count: The number of elements on the active queue at sample time. The active queue holds I/O requests that were sent from the host system and were not yet sent to the wait-on-I/O queue.	PD (11,0)

Figure 6-13 (Part 1 of 2). Twinaxial IOP Data (one for each work station controller)

Field Name	**Description**	**Attributes**
LITWIU	Twinaxial use count: The number of times when the wait-on-I/O queue was sampled and the count was not zero (I/O in progress). If this value is divided by the sample count, the result (times 100) is the percentage of time when I/O is occurring.	PD (5,0)
LISMPL	Sample count: The number of times during the snapshot interval that the various IOP queues were sampled.	PD (5,0)
LIIDLC	Idle counts (see note): The number of times the work station IOP ran an idle loop. This is done when the IOP has no work to perform. This count is used with the idle loop time.	PD (11,0)
LIIDLT	Idle loop time (times 0.01 microsecond): The time (in hundredths of microseconds) to run the idle loop once (see note).	PD (5,0)

Figure 6-13 (Part 2 of 2). Twinaxial IOP Data (one for each work station controller)

Note: The idle loop count and idle loop time are used to calculate the work station IOP utilization as follows:

From the interval time, subtract the product of the idle loop count times the idle loop time. Divide the result by the interval time. For example:

```
IOP utilization = (INTSEC - (LIIDLC x LIIDLT))/ INTSEC
```

File Name: QAPMRESP

Local Work Station Response Time File Entries Figure 6-14 lists the fields in the RI file.

Field Name	Description	Attributes
INTNUM	Interval number: The nth sample interval since the start of the performance monitor job.	PD (5,0)
DTETIM	Interval date (yy/mm/dd) and time (hh:mm:ss): The date and time of the sample interval.	C (12)
INTSEC	Elapsed interval seconds: The number of seconds since the last sample interval.	PD (7,0)
LRIOP	IOP address: An 8-bit field made up of two subfields: IOP bus number: Bits 0, 1, and 2 define the I/O bus number for this IOP. The bus number can range from 0-7. IOP bus address: Bits 3, 4, 5, 6, and 7 define the address of this IOP on this bus. The bus address can range from 0-31.	C (1)
LRWSN	Work station name: The external name that identifies the work station.	C (10)
LRBKT1	Transactions in first response time monitor bracket: The number of transactions from 0 to n seconds for this work station during the *snapshot* interval. The n value is the response time monitor 1 bracket upper limit, and is specified when the performance monitor is started by the STRPRFMON command. A transaction is defined as the time from when the keyboard locked because the Enter key or a function key was pressed to the time when the keyboard is unlocked because the display is refreshed.	PD (7,0)
LRBKT2	Transactions in second response time monitor bracket: The number of transactions between the response time monitor 1 and response time monitor 2 limits.	PD (7,0)
LRBKT3	Transactions in third response time monitor bracket: The number of transactions between the response time monitor 2 and response time monitor 3 limits.	PD (7,0)
LRBKT4	Transactions in fourth response time monitor bracket: The number of transactions between the response time monitor 3 and response time monitor 4 limits.	PD (7,0)
LRBKT5	Transactions in fifth response time monitor bracket: The number of transactions above (longer) than the response time monitor 4 limit.	PD (7,0)
LRWSID	Work station ID: Byte 1 is the port number and byte 2 is the station number of this work station.	C (2)

Figure 6-14. Local Work Station Response Time (one for each station)

Chapter 7. System Values and Network Attributes

This chapter describes system values and network attributes by first giving a summary of the different system values divided by type and then detailed descriptions of the system values. A table showing the network attributes and where more information about each one can be obtained is shown at the end of the chapter.

System values contain specifications that can be used to control or change the overall operation of your system. System values are not objects and cannot be passed as parameter values like CL variables.

A system value can be placed in a CL variable for use in a CL program using the Retrieve System Value (RTVSYSVAL) command. (The *CL Programmer's Guide* contains more information on using CL variables in programs.)

System Value Summary

System values must be enclosed in apostrophes under three conditions:

- If the system value specified is a character string with embedded blanks
- If numeric values or special characters are specified for character type system values
- If the system value is a date or time value

Seven of the system values, QACGLVL, QCHRID, QCMNRCYLMT, QSYSLIBL, QUPSDLYTIM, QUSRLIBL, and QIPLDATTIM may be lists. To separate items in the list, use blanks and enclose the entire list in apostrophes. If there is only one item in the list, you do not need apostrophes. The QUPSDLYTIM system value is a list of two items but only the first item is used when a Change System Value (CHGSYSVAL) command is run. The second item, if specified, is ignored.

Three other values, QCTLSBSD, QSTRUPPGM, and QUPSMSGQ, may be qualified with a library name. If the system values are qualified, use blanks to separate the name and library, and enclose the value in apostrophes (for example, 'QSBSD QLIB'). Apostrophes are necessary only when the library name or *LIBL is specified with the object name. If the library name is not specified or *LIBL is specified for the library, the library list is used to locate the object, and the library where the object is found is stored in the system value.

Note: When object names are specified for system values, the lowercase letters in the names are always changed to uppercase even when they are in apostrophes. This means that you should not use lowercase letters in the names of objects or libraries that you may want to specify on any of the system values.

The following is a summary of the system values and the initial values shipped with the system.

Date and Time System Values

Name	Initial Value	Description	Type	Length
QDATE		System date	Character	5[1] or 6
QYEAR		Year	Character	2
QMONTH		Month of the year (not used for Julian dates)	Character	2
QDAY		Day of the month (day of the year if the system date format is Julian)	Character	2 or 3[1]
QLEAPADJ	0	Leap year adjustment	Decimal	(5 0)
QTIME		Time of day	Character	6, 7, 8, or 9[2]
QHOUR		Hour of the day	Character	2
QMINUTE		Minute of the hour	Character	2
QSECOND		Second of the minute	Character	2

[1] For Julian dates

[2] For tenths, hundredths, and thousandths of a second

Editing System Values

Name	Initial Value	Description	Type	Length
QCURSYM	'$'	Currency symbol	Character	1
QDATFMT	MDY	Date format	Character	3
QDATSEP	'/'	Date separator	Character	1
QDECFMT	'ƀ'	Decimal format	Character	1

System Control System Values

Name	Initial Value	Description	Type	Length
QABNORMSW	'0'	Previous end of system indicators. ('0' means previous end was normal. '1' means previous end was abnormal.) Cannot be changed.	Character	1
QAUTOCFG	'1'	Autoconfiguration indicator. ('0' means autoconfiguration is off. '1' means autoconfiguration is on.)	Character	1
QCHRID	'101 37'	Default graphic character set and code page used for displaying or printing data.	Character	20
QCMNRCYLMT	'0 0'	Provides recovery limits for system communications recovery.	Character	20
QCONSOLE	'DSP01'	Console name. Cannot be changed.	Character	10
QCTLSBSD	'QBASE QGPL'	Controlling subsystem name.	Character	20
QDBRCVYWT	'0'	Data base recovery indicator. ('0' means wait. '1' means do not wait.)	Character	1
QDEVNAMING	'*NORMAL'	Indicates the device naming convention. ('*NORMAL' means follow AS/400 standards. '*S36' means follow System/36 standards.)	Character	10
QIGC	'0'	Indicates if the DBCS version of the system is installed. ('1' means the DBCS version is installed. '0' means the DBCS version is not installed.) Cannot be changed.	Character	1
QIPLDATTIM	'*NONE'	Date and time to automatically IPL the system.	Character	20
QIPLSTS	'0'	IPL status indicator. ('0' specifies operator panel IPL. '1' specifies auto-IPL after power is restored. '2' specifies start IPL again. '3' specifies auto-IPL at some TIME OF DAY. '4' specifies remote IPL.) Cannot be changed.	Character	1
QIPLTYPE	'0'	Indicates type of IPL to perform. ('0' means unattended IPL. '1' means attended IPL with service screens.)	Character	1
QKBDTYPE	'USB'	Specifies a language character set for the keyboard.	Character	3
QMAXSIGN	'15'	Maximum number of invalid sign-on attempts allowed.	Character	6
QPFRADJ	'1'	Performance adjustment. ('0' means no performance adjustment. '1' means performance adjustment at IPL.)	Character	1

Name	Initial Value	Description	Type	Length
QPRTDEV	'PRT01'	Default printer device description.	Character	10
QPWRDWNLMT	'300'	Maximum amount of time (in seconds) allowed for PWRDWNSYS *IMMED.	Decimal	(5 0)
QPWRRSTIPL	'0'	Automatic IPL after power restored allowed. ('0' means no auto-IPL after power restored. '1' means auto-IPL after power restored.)	Character	1
QRMTIPL	'0'	Remote power on and IPL indicator. ('0' means remote power on and IPL is not allowed. '1' means remote power on and IPL is allowed.)	Character	1
QSCPFCONS	'1'	IPL console indicator. ('1' means to switch to unattended IPL if console problems occur during IPL. '0' means end system.)	Character	1
QSECURITY	'10'	Indicates security level. ('10' means physical security only. '20' means password security only. '30' means password security and resource security.)	Character	2
QSPCENV	'*NONE'	Indicates default special environment. ('*NONE' means no special environment. '*S36' means System/36 environment.)	Character	10
QSRLNBR		System serial number. Cannot be changed.	Character	8
QSTRPRTWTR	'1'	Indicates if print writers should be started. ('0' means print writers not started. '1' means start print writers.) Cannot be changed.	Character	1
QSTRUPPGM	'QSTRUP QSYS'	Startup program name called from autostart job in the controlling subsystem.	Character	20
QUPSDLYTIM	'*CALC'	Uninterruptible power supply delay time.	Character	20
QUPSMSGQ	'QSYSOPR QSYS'	Message queue for uninterruptible power supply messages.	Character	20

Library List System Values

Name	Initial Value	Description	Type	Length
QSYSLIBL	'QSYS QHLPSYS QUSRSYS'	System part of the library list.	Character	150
QUSRLIBL	'QGPL QTEMP'	User part of the library list.	Character	250

Allocation System Values

Name	Initial Value	Description	Type	Length
QACTJOB	20	Initial number of active jobs to allocate storage for.	Decimal	(5 0)
QADLACTJ	10	Additional number of active jobs to allocate storage for.	Decimal	(5 0)
QADLSPLA	2048	Additional storage for extending spooling control block (bytes).	Decimal	(5 0)
QADLTOTJ	10	Additional total number of jobs to allocate storage for.	Decimal	(5 0)
QJOBMSGQSZ	16	The size used to create job message queues (K bytes).	Decimal	(5 0)
QJOBMSGQTL	24	Maximum reinitialization size of job message queue (K bytes).	Decimal	(5 0)
QJOBSPLA	1536	Initial size of spooling control block for a job (bytes).	Decimal	(5 0)
QTOTJOB	30	Initial total number of jobs to allocate storage for.	Decimal	(5 0)

Message and Logging System Values

Name	Initial Value	Description	Type	Length
QACGLVL	*NONE	Accounting level.	Character	80
QPRTTXT	blanks	Up to 30 characters of text that can be printed at the bottom of the form.	Character	30

Name	Initial Value	Description	Type	Length
QHSTLOGSIZ	5000	Maximum number of records for each version of the history log.	Decimal	(5 0)
QSRVDMP	*DMPUSRJOB	Control for requesting dumps: no jobs, system jobs, user jobs, or all jobs.	Character	10

Storage System Values

Name	Initial Value	Description	Type	Length
QBASACTLVL	6	Activity level of base storage pool.	Decimal	(5 0)
QBASPOOL	500	Minimum size of base storage pool (K bytes).	Zoned	(10 0)
QMAXACTLVL	100	Maximum activity level of the system.	Decimal	(5 0)
QMCHPOOL	1500	Machine storage pool size (K bytes).	Zoned	(10 0)

Displaying a System Value

You use the Display System Value (DSPSYSVAL) command to display a system value. The DSPSYSVAL command description in the *CL Reference* contains a detailed description of the system value display.

Changing a System Value

Not all system value changes are used by the system immediately when the Change System Value (CHGSYSVAL) command is issued. For some values, such as QCTLSBSD, the change takes effect the next time the system is IPLed.

The following CHGSYSVAL command changes the value of the system value QUSRLIBL:

```
CHGSYSVAL SYSVAL(QUSRLIBL) VALUE('DSTPRODLB QGPL QTEMP')
```

This system value defines the system default for the user part of the initial library list. The initial library list for new jobs is redefined and DSTPRODLB is placed before QGPL and QTEMP. The change takes effect immediately for jobs started after the CHGSYSVAL command is run (but not for jobs that are already active).

How the value is entered on the VALUE parameter is dependent on the type of system value: character or numeric. The following rules for specifying the VALUE parameter apply to system values.

- If a character string contains other than alphameric characters, a period, or a comma, it must be enclosed in apostrophes.
- If a list of values is specified as the new value, the list must be enclosed in apostrophes and separated by blanks.
- If a numeric value or special character is specified for a character type system value, it must be enclosed in apostrophes.
- Date and time values must be enclosed in apostrophes and cannot contain separators.
- All qualified names in system values should be specified as 'obj-name lib-name'.

All numeric type system values must be specified as whole numbers greater than or equal to 0 and not exceeding 32 767. If a numeric system value is specified by using a CL variable, the variable must be decimal. A numeric value is not enclosed in apostrophes unless it is specified for a character type system value. Each system value has its own set of limitations on values. If a CL variable is used to specify the value for a character type system value, the CL variable must be character type and may be of any length. Values are padded with blanks or truncated, as necessary.

Detailed Description of System Values

System values are provided by IBM. You cannot create them. The following is a list of the IBM-supplied system values.

Date and Time System Values

Date and time system values must always be enclosed in apostrophes.

If you change a system value during any operation that measures the length of time, a negative value may be set if the end time is less than the start time.

QDATE: System date. Its value can be set from the IPL options display and is updated when the system value QTIME reaches midnight (000000). QDATE is a 6-character value (5 characters for Julian) composed of the following values (each of which can be individually referred to). The format of the date is as specified in the system value QDATFMT. A change made to this value takes effect immediately.

- *QYEAR.* Year. Its value can range from 0 through 99. When you specify the date in CL commands, you can specify only 2 digits for the year in the date format. The system assigns the first 2 digits for the year (xx) based on the following rules:
  - If YY is equal to or greater than 40, the system assigns a value of 19 (19xx).
  - If YY is less than 40, the system assigns a value of 20 (20xx).

  You should, however, be careful when specifying the date on the PERIOD parameter; an incorrect date will cause the command to fail. For example, if you specify PERIOD(090929 *CURRENT) on the Display Log (DSPLOG) command, the command fails because the system interprets the value 29 (specified for the year) as 2029 instead of 1929. Because 2029 is later than the current date (19xx), the value is rejected, and the command fails.
- *QMONTH.* Month of the year. Its value can range from 1 through 12. (Not valid for Julian.)
- *QDAY.* Day of the month. Its value must be a valid day of the specified month and year. For Julian dates only, QDAY is a 3-character value (001 through 366 for Julian).
- *QLEAPADJ.* Leap year adjustment. This system value is used to adjust the system calendar algorithm for the leap year in different calendar systems. If your calendar year agrees with what is used in the Gregorian calendar system, then this system value should be zero. If your calendar year differs from the Gregorian, you may need to adjust the system calendar algorithm to account for the leap year of that calendar year you are using. To make the adjustment, divide the leap year in your calendar system by 4; then set QLEAPADJ to the value of the remainder.

  Example: The Gregorian calendar year of 1984 is the year 73 in the Republic of China calendar. Because 73 is a leap year for the Republic of China, you need to divide 73 by 4; this leaves a remainder of 1. Therefore, to adjust the system calendar algorithm for the Republic of China, specify a 1 for the QLEAPADJ system value.

  Changing the QLEAPADJ system value does not change the system clock and job dates of active jobs, but it may change the QDATE system value.

QTIME: Time of day. Its value can be set from the IPL options display. QTIME is a 6- to 9-character value depending on the resolution required. A length of 6 characters provides 2 characters each for hours, minutes, and seconds. As the length increases, the time resolution becomes more precise (tenths of a second in position 7, hundredths of a second in position 8, and thousandths of a second in position 9). A change made to this value takes effect immediately.

- *QHOUR*. Hour of the day. Its value can range from 00 through 23.
- *QMINUTE*. Minute of the hour. Its value can range from 00 through 59.
- *QSECOND*. Second of the minute. Its value can range from 00 through 59.

The Retrieve System Value (RTVSYSVAL) command allows the CL variable to be greater than or equal to 6 characters. If the CL variable is larger than 9 characters, then the first 9 characters contains hours, minutes, seconds, and milliseconds and the others are padded with blanks. The Change System Value (CHGSYSVAL) command can be used to set milliseconds to zero on a change to QTIME, QHOUR, QMINUTE, or QSECOND. Milliseconds cannot be specified on any system value change. The Display System Value (DSPSYSVAL) does not display milliseconds.

The following table shows the various values provided by the CL character variable.

Length of CL Character Variable	Meaning
<6	Invalid.
6	Hours, minutes, and seconds.
7	Hours, minutes, seconds, and tenths of a second.
8	Hours, minutes, seconds, tenths of a second, and hundredths of a second.
9	Hours, minutes, seconds, tenths of a second, hundredths of a second, and thousandths of a second.
>9	Hours, minutes, seconds, tenths of a second, hundredths of a second, thousandths of a second, and padded with blanks.

QCURSYM: Currency symbol. This system value is used to validate the currency symbols specified in the DDS keywords EDTWRD and EDTCDE. QCURSYM is a 1-character value and can be any character except blank, -, &, *, or 0.

QDATFMT: System date format. This system value is 3 characters that can be YMD, MDY, DMY, or JUL (Julian format). (Y = year; M = month; D = day.) This system value is used for the following:

- The default value for the DATFMT job attribute
- To determine the format in which a date can be specified on the IPL options prompt

For more information on DATFMT, see the RTVJOBA and CHGJOB commands in the *CL Reference.*

QDATSEP: Character separator for dates. This system value is used as the date separator for the following:

- The default value for the DATSEP job attribute
- In the date that can be specified on the IPL options prompt

QDATSEP is a 1-character value and can be /, -, period (.), or comma (,).

The RTVJOBA and CHGJOB commands in the *CL Reference* contain more information on DATSEP.

QDECFMT: Decimal format. This is a 1-character system value used for the following:

- To determine the type of zero suppression and decimal point character used by DDS edit codes 1 through 4 and A through M
- To determine the decimal point character for decimal input fields on displays

QDECFMT must be one of the following characters:

- ƀ: Use a period for a decimal point, use a comma for a three-digit grouping character, and zero-suppress to the left of the decimal point.
- J: Use a comma for a decimal point, and use a period for a three-digit grouping character. The zero-suppression character is in the second position (rather than the first) to the left of the decimal notation. Thus balances with zero values to the left of the comma are written with one leading zero (0,04). The J entry also overrides any edit codes that might suppress the leading zero.
- I: Use a comma for a decimal point, use a period for a three-digit grouping character, and zero-suppress to the left of the decimal point.

System Control System Values

QABNORMSW: Previous end of system indicator. Specifies if the previous end of system was normal or abnormal. The previous end was normal if it is the result of a successful PWRDWNSYS command and the ENDJOBABN was not used. If you end a job using the ENDJOBABN command, the next system end will be abnormal. You cannot change QABNORMSW; it is set by the system. QABNORMSW is a 1-character value that can be '0' (normal) or '1' (abnormal). You can refer to this value in user-written recovery programs.

QAUTOCFG: Autoconfiguration indicator. Specifies if devices that are dynamically added to the system should be configured automatically.

- '0': Autoconfiguration is off. You will have to manually configure any new local controllers or devices that you add to your system.
- '1': Autoconfiguration is on. The system will automatically configure any new local controllers or devices that are added to your system. The operator will receive a message indicating the changes to the system's configuration.

QCHRID: Default character set and code page. This system value specifies the character set and code page to be used when CHRID(*SYSVAL) is specified for the CL commands that create, change, or override display files, display device descriptions, and printer files. In general, the value you specify should be one of the values shown for the CHRID parameter on the CRTDSPF, CRTDEVD, or CRTPRTF command in the *CL Reference*. Specify a value that reflects the language used at most of the devices attached to your system. The *Data Management Guide* contains more information in the sections on using alternative graphic character sets and code pages.

QCHRID is a single 20-character value with two parts:

- Character set identifier: This part identifies the character set to use. This identifier must be in the range of 1 through 32 767.
- Code page identifier: This part identifies the code page to use. This identifier must be in the range of 1 through 32 767.

The QCHRID system value is retrieved as a single character value; the first 10 characters contain the character set identifier right-adjusted. For example, the value 101 would be retrieved as '0000000101'. The last 10 characters contain the code page identifier right-adjusted. For example, the value 37 would be retrieved as '0000000037'. The QCHRID system value is displayed in the same format as other list type system values.

QCMNRCYLMT: Communications recovery limits. The value of QCMNRCYLMT is a 20-character list containing two values:

- Count limit: The number of recovery attempts (0-99) to be made by the system before an inquiry message is sent to the system operator.
- Time interval: The time period (0-120 minutes) before the system will send an inquiry message to the system operator if the count limit is reached.

Both values are integer values and must be specified in a quoted string. A blank must separate the count limit from the time interval. The possible recovery actions are:

Count Limit	Time Interval	Action
0	0	No recovery
0	1 through 120	No recovery
1 through 99	0	Infinite recovery
1 through 99	1 through 120	Count and time recovery

If your AS/400 system is attached to a ROLM CBX, the recovery attempts value should never be 0. Recovery attempts are necessary for the AS/400 system to establish a connection using the ROLM CBX's inbound modem pool.

A change to the QCMNRCYLMT system value does not affect a varied on device, but is in effect when a device is varied on. The *Communications Programmer's Guide*. contains more information on communications recovery.

The QCMNRCYLMT system value is retrieved as a 20-character value: the first 20 characters contain the count limit right adjusted, for example, the value 7 would be retrieved as '0000000007'. The last 10 characters contain the time interval right adjusted, for example, the value 117 would be retrieved as '0000000117'. The QCMNRCYLMT is displayed in the same format as other list type system values.

QCONSOLE: Console name. Specifies the name of the display device that is the console. You cannot change this system value.

QCTLSBSD: Controlling subsystem. The value of QCTLSBSD is a 20-character list of up to two 10-character values in which the first is the subsystem description name and the second is the library name. The list must be specified as a quoted string separating the subsystem description name and the library name by a blank if the library is specified (for example, 'QSBSD QLIB'). Apostrophes are necessary only when the library name or *LIBL is specified with the subsystem name. If the library name is not specified or *LIBL is specified for the library, the library list is used to locate the subsystem description, and the library where the subsystem description is found is stored in the system value. If *LIBL is specified for the library name for a change made from selecting 'Define or Change System at IPL Menu' on the IPL options display, during IPL, QGPL is the only library in the library list (*LIBL).

A change to this value takes effect at the next IPL unless the change is made on the IBM-supplied configuration menu during IPL, in which case a change to this value takes effect immediately. If this subsystem description cannot be used (for example, it is damaged), the backup subsystem description QSYSSBSD in the library QSYS can be used. A subsystem description specified as the controlling subsystem cannot be deleted or renamed once the system is fully operational. Refer to "Creating Another Controlling Subsystem" on page 2-47 in Chapter 2, "Using Work Management

Functions" on page 2-1 for other items to consider before changing the value of QCTLSBSD.

The subsystem you will be signed on to is displayed in the upper right corner of the sign-on prompt. During an IPL, if the QCTLSBSD system value is changed by selecting 'Define or Change System' on the IPL options display, the user will be signed on a different subsystem than the one that was displayed on the Sign-On display. This is because the change takes effect immediately.

QDBRCVYWT: Data base recovery wait indicator. Indicates when recovery of files created with the RECOVER(*AFTIPL) option is performed during an unattended IPL. QDBRCVYWT is a 1-character value that can be:

- '0': Do not wait for data base recovery to complete.
- '1': Wait for data base recovery to complete before completing the IPL.

If QDBRCVYWT is '1', then, during an unattended IPL, files which were created with the RECOVER(*AFTIPL) option will be treated as if they had been created with the RECOVER(*IPL) option. This value does not affect recovery of files that are created with RECOVER(*IPL) or RECOVER(*NO). A change to this value takes effect during the next IPL in unattended mode. QDBRCVYWT has no effect during an attended IPL. This value is related to the RECOVER parameter on the Create Physical File (CRTPF) and Create Logical File (CRTLF) commands.

QDEVNAMING: Device naming convention. Specifies what naming convention is used when the system automatically creates device descriptions. You can change, display, and retrieve this value. QDEVNAMING is a 10-character value that can be:

- '*NORMAL': Naming conventions should follow AS/400 standards.
- '*S36': Naming conventions should follow System/36 standards.

The following shows an example of the normal naming convention and the S36 naming convention:

Device	Normal	S36
Display Stations	DSP01, DSP02	W1, W2
Diskette Drive	DKT01	I1

For more information on the device naming conventions, see the *Device Configuration Guide*.

QIGC: DBCS character indicator. Specifies if the DBCS version of the system is installed. You cannot change QIGC; it is set by the system. You can refer to this system value in an application program. QIGC is a 1-character value that can be:

- '0': The DBCS version is not installed.
- '1': The DBCS version is installed.

QIPLDATTIM: Date and time to automatically IPL the system. Specifies a date and time when an automatic IPL should occur. The default value *NONE indicates that no timed automatic IPL is desired. QIPLDATTIM is a 20-character list of two 10-character values. The first value is the date when the timed IPL should occur. The date is in QDATFMT format with no date separators. The date cannot be more than 11 months after the current date. The second value is the time. The seconds portion of the time value must be specified, but is ignored. If the date and time has already occurred when the system is powered down or the system is running when the date and time occur, no IPL is performed. The value of QIPLDATTIM does not go back to *NONE after the IPL occurs or the date and time passes, but remains until it is changed. The following example shows how to change the IPL date and time to September 10, 1987, at 9:00 a.m. (QDATFMT is MDY).

```
CHGSYSVAL SYSVAL(QIPLDATTIM) VALUE('091087 090000')
```

QIPLSTS: IPL status indicator. Indicates what form of IPL has occurred. There is an indicator value for each form of IPL. You can refer to this value in your recovery programs, but you cannot change it.

- '0': Operator panel IPL. IPL occurred when requested from the operator panel.
- '1': Auto-IPL after power restored. IPL occurred automatically when power was restored after a power failure. This is enabled by the QPWRRSTIPL system value.
- '2': Restart IPL. IPL occurred when the Power Down System (PWRDWNSYS) command with RESTART(*YES) was issued.
- '3': Time-of-day IPL. IPL occurred automatically on the date and time set on the QIPLDATTIM system value.
- '4': Remote IPL. Remote IPL occurred. This is enabled by the QRMTIPL system value.

QIPLTYPE: Indicates type of IPL to perform. Specifies the type of IPL performed when the system is powered on manually with the key in the normal position. QIPLTYPE is a 1-character value that can be:

- '0': Unattended. No screens are shown during IPL. The normal sign-on screen is shown when the IPL is complete.
- '1': Attended with dedicated service tools. All dedicated service tools functions are available along with the full set of IPL screens.

QKBDTYPE: Specifies the language character set for the keyboard. Based on the language value specified at install time, OS/400 will put the appropriate keyboard value into the this system value. QKBDTYPE is a 3-character value that can be as follows:

QKBDTYPE Value	Language/Country
AGB	Austria/Germany
AGI	Austria/Germany Multinational
BLB	Belgium
BLI	Belgium Multinational
CAB	Canadian French
CAI	Canadian French Multinational
CLB	Arabic X/Basic
CYB	Cyrillic
DMB	Denmark
DMI	Denmark Multinational
FNB	Finland/Sweden
FNI	Finland/Sweden Multinational
FAB	France (Azerty)
FAI	France (Azerty) Multinational
FQB	France (Qwerty)
FQI	France (Qwerty) Multinational
GKB	Greece

ICB	Iceland
ICI	Iceland Multinational
INB	International
INI	International Multinational
ITB	Italy
ITI	Italy Multinational
JEB	Japan (English)
JEI	Japan (English) Multinational
JKB	Japan (Kanji)
KAB	Japan (Katakana)
KOB	Korea
NCB	Hebrew
NEB	Netherlands
NEI	Netherlands Multinational
NWB	Norway
NWI	Norway Multinational
PRB	Portugal
PRI	Portugal Multinational
RCB	Simplified Chinese
ROB	Latin 2
SPB	Spain
SPI	Spain Multinational
SSB	Spanish Speaking
SSI	Spanish Speaking Multinational
SWB	Sweden
SWI	Sweden Multinational
SFB	Switzerland/French
SFI	Switzerland/French Multinational
SGB	Switzerland/German
SGI	Switzerland/German Multinational
TAB	Traditional Chinese
THB	Thailand
TKB	Turkey
UKB	United Kingdom
UKI	United Kingdom Multinational
USB	United States/Canada
USI	United States/Canada Multinational
YGI	Yugoslavia Multinational

QMAXSIGN: Maximum number of invalid sign-on attempts allowed. Invalid sign-on attempts arise from any of the following circumstances:

- Invalid user name (if secured system).
- Invalid password (if secured system).
- The user profile does not have authority to the device from which the user name was entered (if secured system).

Note: A sign-on attempt will not be counted as an invalid attempt if passwords are required and the user profile has a password of *NONE specified. The user will receive a message saying that no password is associated with the user profile. It also will not be counted as an invalid attempt if (1) the program or menu are not valid names, (2) if the user name is not a valid name on a non-secured system, or (3) if the current library specified is not found.

If the QMAXSIGN value maximum is reached, the work station device is varied off and a message is sent to the QSYSMSG message queue if it exists; otherwise, it is sent to QSYSOPR. The device must be varied on again before a user can sign on. If the controlling subsystem is in the restricted state (so that only one device in it can be used) and the maximum is reached, the system is ended and control panel lights on the control panel turn on to indicate that you must perform an IPL. QMAXSIGN is character. The lower limit for QMAXSIGN is 1. The initial value is '15'. To change the QMAXSIGN system value, the user must have security officer (QSECOFR) authority, which can be from a user profile or a group profile or it can be adopted by a program.

When the number of invalid sign-on attempts is one less than QMAXSIGN, message CPF1116 is sent to the display to warn the user that if another invalid attempt is made the device will become unusable. In some environments, it is not desirable to have this message appear. You cannot prevent this message from being sent, but you can change the message text (by using the CHGMSGD command) to the same text as CPF1107 *Password not valid for system.* Refer to CPF1107 for the values entered for the MSG and SECLVL parameters.

QPFRADJ: Performance adjustment. Indicates whether the system should adjust values for system pool sizes and activity levels as follows:

- '0': No performance adjustment. The existing values for system pool sizes and activity levels are used.
- '1': Performance adjustment at IPL. The values for system pool sizes and activity levels are calculated and changed during IPL. The following values are changed:
  - QMCHPOOL system value
  - QBASACTLVL system value (if QGPL/QBASE or QGPL/QCTL is the controlling subsystem)
  - Pool 2 of QGPL/QSPL subsystem description
  - Pool 2 of QGPL/QBASE subsystem description (if QGPL/QBASE is controlling subsystem)
  - Pool 2 of QGPL/QINTER subsystem description (if QGPL/QCTL is controlling subsystem)

The value of QPFRADJ should be set to '0' if you have set any of these values.

The performance adjustment is only done once during IPL; it does not dynamically change with the configuration or workload. After your configuration has stabilized, QPFRADJ can be changed to '0' to save time during IPL. Be sure to do at least one IPL after your configuration is set.

QPRTDEV: Default printer device description. Specifies the default printer for the system. This value will take effect when a spooled file is opened. If the specified device description exists, it must be a printer device description. The initial value is PRT01.

QPWRDWNLMT: Maximum amount of time an immediate power down can take before processing is ended (abnormal end). This is the time used to wait for power down to complete normally after either of the following happens:

- A Power Down System (PWRDWNSYS) command with an OPTION(*IMMED) parameter is entered.
- A PWRDWNSYS command with an OPTION(*CNTRLD) parameter was entered and the time specified on the DELAY parameter has ended.

The QPWRDWNLMT value is ignored when either of these commands is entered after a power failure has occurred on a system with the power warning feature installed (see the *Backup and Recovery Guide* for more details).

A change to this value takes effect when a PWRDWNSYS command is entered. QPWRDWNLMT is numeric. The lower limit for QPWRDWNLMT is 0. If the value is set to 0 and a PWRDWNSYS command with OPTION(*IMMED) is entered, processing is ended immediately (abnormal end).

QPWRRSTIPL: Automatic IPL is allowed after power restored. Specifies if the system should automatically IPL when utility power is restored after a power failure. QPWRRSTIPL is a 1-character value that can be:

- '0': Auto-IPL is not enabled.
- '1': Auto-IPL is enabled.

QRMTIPL: Remote IPL indicator. Specifies if remote power on and IPL can be started over a phone line. QRMTIPL is a 1-character value that can be:

- '0': Remote IPL is not enabled.
- '1': Remote IPL is enabled.

QSCPFCONS: SCPF console indicator. Indicates if the IPL is to continue unattended or to end when the console is not operational when trying an attended IPL. QSCPFCONS must have been set before the current IPL. QSCPFCONS is a 1-character value that can be '0' (end) or '1' (continue unattended). It should be '0' if there are no work stations other than the console on the system or if the controlling subsystem supports only the console and does not start other subsystems that support other work stations.

QSECURITY: Security indicator. Specifies the level of security on the system. To change this system value, the user must have security officer (QSECOFR) authority, which can be from a user profile or a group profile or it can be adopted by a program. The QSECURITY system value can be changed at any time using the CHGSYSVAL command, but the change to the security level does not take effect until the next IPL as follows:

Attended IPL: The security level required to sign on for the IPL will stay the same as what was in effect before a system IPL was performed. Changes will be made during the IPL but after the sign-on for the IPL.

Note: During an attended IPL, you can also change the QSECURITY system value by selecting "Define or Change System" on the IPL options display and the change will take effect when the IPL is done. This change would override any change that was requested before the IPL.

Unattended IPL: Previously requested changes to the security level will be made during the IPL.

QSECURITY is a 2-character value that can be:

- '10': The system does not require a password at sign-on. The user has access to all system resources. Special authorities can be added or removed from a user profile using the CRTUSRPRF or CHGUSRPRF command.
- '20': The system will require a password at sign-on. The user has access to all system resources.
- '30': The system will require a password at sign-on. The user must have authority to access all system resources.

  All user profiles will have special authorities granted or revoked based on their user class when changing from a level 10 or 20 system.

QSPCENV: Special environment indicator. Specifies the system environment used as the default for all user profiles. This value will not be saved in the user profile. It will be checked at sign-on time. QSPCENV is a 10-character value that can be:

- '*NONE': You will enter the AS/400 system environment at job start time.
- '*S36': You will enter the System/36 environment at job start time.

QSRLNBR: System serial number. You cannot change QSRLNBR; it is retrieved from the machine configuration record by the system when installing OS/400. You can display QSRLNBR, or you can retrieve this value in user-written programs. QSRLNBR is an 8-character value.

Note: The system serial number is also used to set the system name. There is a discussion of the Change Network Attributes (CHGNETA) command in the *Communications Programmer's Guide*.

QSTRUPPGM: Start-up program name called from autostart job in the controlling subsystem. This program performs setup functions such as starting subsystems and printers. This system value can only be changed by the security officer or a profile with security officer authority. QSTRUPPGM is a 20-character value that can be:

- 'QSTRUP QSYS': The program specified will be run as a result of a transfer control to it from the autostart job in the controlling subsystem.
- '*NONE': The autostart job will end normally without calling a program.

The QSTRUP IBM-supplied program does the following:

- Starts the QSPL subsystem for spool work.
- Releases the QS36MRT and QS36EVOKE job queues if they were held (these are used by the System/36 environment).
- Starts all print writers unless user said not to on the IPL Options display.
- If the controlling subsystem is QCTL, it starts the QINTER, QBATCH, and QCMN subsystems.

QSTRPRTWTR: Indicates if print writers are started. This system value is either set by the system at IPL time or by the user on the IPL Options display. This value can only be displayed or retrieved. The IBM-supplied startup program QSTRUP uses this system value to determine whether to start print writers. QSTRPRTWTR is a 1-character value that can be:

- '0': Do not start print writers.
- '1': Start print writers.

QUPSDLYTIM: Uninterruptible power supply (UPS) delay time. The QUPSDLYTIM system value indicates the amount of time that should elapse before the system automatically powers down, following a power failure. This system value is only meaningful if your system has the battery power unit or has an uninterruptible power supply (UPS) attached. The allowed values are:

Value	Description
'*CALC'	The appropriate wait time (in seconds) will be calculated. (This should only be used if you have the battery power unit.) **Notes:** 1. The calculated value is only appropriate for the 9404 user (although there is no restriction on its use by a 9406 user). 2. The calculated value is based on the amount of devices and the number of changed pages in main storage. 3. The calculated value may have to be adjusted by the user if there has been a series of power outages, or if the batteries are not fully charged.
'*NOMAX'	The system will not start any action on its own.
'0'	Automatic system power down when system utility power fails.
'1'-'99999'	Delay time specified in seconds before the system powers down.

The initial value is *CALC. A change to this value takes effect immediately. For more information on the uninterruptible power supply feature, see the *Backup and Recovery Guide*.

The QUPSDLYTIM system value is in the form of a two-item list. The first item is the value the user specified on the CHGSYSVAL command. The second item is the delay time, which is either what the user specified or, if *CALC was specified, is the calculated delay time.

You specify only the first item in the list. The second item, if specified, will be ignored. If you want to retrieve the QUPSDLYTIM system value, you would retrieve it into a CHAR 20 variable. The first 10 characters would be what you specified. The second 10 characters would be the delay time value (calculated by the machine or specified by you) you would use in your program.

10 Characters	10 Characters
0000000340	0000000340
*NOMAX	*NOMAX
*CALC	0000000120

The following examples show what the user could enter and how the value would be displayed:

Example 1

User enters:

```
CHGSYSVAL QUPSDLYTIM '340'
```

Display shows:

Value
0000000340
0000000340

Example 2

User enters:

```
CHGSYSVAL QUPSDLYTIM '*NOMAX'
```

Display shows:

Value
*NOMAX
*NOMAX

Example 3

User enters:

```
CHGSYSVAL QUPSDLYTIM '*CALC'
```

Display shows:

Value
*CALC
0000000120

QUPSMSGQ: Name of a message queue that is to receive uninterruptible power supply messages. If the message queue is not the system operator message queue ('QSYSOPR QSYS'), then all uninterruptible power supply messages are also sent to the QSYSOPR message queue. This system value is meaningful only if your system has the battery power unit feature and has an uninterruptible power supply attached.

When a change in power activates the uninterruptible power supply, this message queue receives the uninterruptible power supply activated message (CPF1816). If the QUPSDLYTIM is '*NOMAX', the following conditions must be met or the system will begin an immediate power down.

- The message queue specified in the QUPSMSGQ system value must exist.
- If the message queue is a work station message queue (or QSYSOPR), it must be in break or notify mode.
- If the message queue is not a work station message queue, it must be allocated by a job.

For all other uninterruptible power supply messages, the message queue does not have to be allocated, or in break or notify mode. If the system value QUPSMSGQ does not contain the name of a valid message queue, a message will be sent to QSYSOPR indicating the notification failure, and the system will continue processing.

For more information on the uninterruptible power supply feature, see the *Backup and Recovery Guide*.

Library List System Values

QSYSLIBL: System part of library list. Its value is a 150-character list, enclosed in apostrophes, of library names with each name up to 10 characters in length, separated by blanks. The list can contain as many as 15 names. Apostrophes are necessary only when more than one library name is specified (for example, 'SYSLIB1 SYSLIB2'). The libraries in the system part are searched for an object before any libraries in the user part. A change to this value takes effect immediately (when the next job starts). A library specified as part of the library list cannot be deleted or renamed once the system is fully operational. To change the QSYSLIBL system value, you must have security officer (QSECOFR) authority, which can be from a user profile, a group profile, or it can be adopted by a program.

QUSRLIBL: Default for user part of library list. Its value is a 250-character list, enclosed within apostrophes, of library names with each name up to 10 characters in length, separated by blanks. The list can contain as many as 25 names. Apostrophes are necessary only when more than one library name is specified. For example, 'USERLIB1 USERLIB2'. The libraries in this part are searched for an object after the libraries in the system part and also after the product library and current library entries. However, if the user part of the library list is specified for a job through a Change Library List (CHGLIBL) command or the job description, or is changed through the Add Library List Entry (ADDLIBLE) or Remove Library List Entry (RMVLIBLE) commands, QUSRLIBL is no longer used for that job. A change to this value takes effect immediately (when the next job starts). A library specified as part of the library list cannot be deleted or renamed once the system is fully operational.

Allocation System Values

QACTJOB: Initial number of active jobs for which auxiliary storage is to be allocated during IPL. An active job is a job that has started running but not ended. The amount of auxiliary storage allocated for each active job is approximately 110K. This storage is in addition to the storage allocated using the system value QTOTJOB. QACTJOB is numeric. The lower limit for QACTJOB is 1. A change to this value takes effect at the next IPL unless the change is made on the IBM-supplied configuration menu, in which case it is used for the current IPL. QACTJOB may be changed to a smaller value during IPL if the combination of allocation system values requires more auxiliary storage than is available to start the system. If this happens, you will be notified.

A reasonable value to assign to QACTJOB is your estimate of the number of active jobs on a typically heavy use day. This can be done by viewing the active jobs field on the active jobs display (WRKACTJOB command). Both user and system jobs are included in the count on this display and should be considered when assigning a value to QACTJOB.

QADLACTJ: Additional number of active jobs for which auxiliary storage is to be allocated when the initial number of active jobs (the system value QACTJOB) is reached. This auxiliary storage is allocated whenever the number of active jobs exceeds that for which storage has already been allocated. The amount of storage allocated for each job is approximately 110K. QADLACTJ is numeric. The lower limit for QADLACTJ is 1. A change to this value takes effect immediately. QADLACTJ may be changed to a smaller value during IPL if the combination of allocation system values requires more auxiliary storage than is available to start the system. If this happens, you will be notified.

An initial value of 10 is reasonable to assign to QADLACTJ. The number should not usually be set too close to 1, because this causes frequent interruption on a day when many additional jobs are needed. The number should not usually be set too high because the time required to add additional storage should be minimized.

QADLSPLA: Additional storage to extend the spooling control block. This storage is allocated when the number of spooled files exceeds the initial storage assigned by the QJOBSPLA system value or the last extension from QADLSPLA. This value affects auxiliary storage, but also affects processing performance each time an extension is needed. For this reason, a relatively high default of 2048 bytes is used to allow an extension for approximately four or five spooled files. No additional bytes are added to the value you supply, but the value will be rounded up (if needed) to the next 512-byte page boundary.

If you have sufficient auxiliary storage, a value of 4096 is a good choice to allow for approximately nine or ten spooled files per extension. The next logical choice would be 8192. Set the QADLSPLA value so that it rounds up (if needed) to a 2K, 4K, or 8K value.

QADLSPLA is numeric. The lower limit is 1024 bytes. A change to this value takes effect for the next time an extension is needed.

QADLTOTJ: Additional number of jobs for which auxiliary storage is to be allocated when the initial number of jobs (the system value QTOTJOB) is reached. This auxiliary storage is allocated whenever the number of jobs exceeds that for which storage has already been allocated. The amount of storage allocated for each job is approximately the sum of the QJOBSPLA and QJOBMSGQSZ system values.

Note: The system value QJOBSPLA is measured in bytes and the system value QJOBMSGQSZ is measured in K bytes.

QADLTOTJ is numeric. The lower limit for QADLTOTJ is 1. A change made to this value takes effect immediately. QADLTOTJ may be changed to a smaller value during IPL if the combination of allocation system values requires more auxiliary storage than is available to start the system. If this happens, you will be notified.

An initial value of 10 is reasonable to assign to QADLTOTJ. The number should not usually be set too close to 1 as this will cause excessive interruption on a day when many additional jobs are needed. The number should not usually be set too high because the time required to add additional storage should be minimized.

QJOBMSGQSZ: Initial size of the job message queue. To determine the approximate size of a message queue, use 100 bytes for each message on the queue. The allocated space is automatically extended when the initial size is exceeded. However, you should try to allocate the correct amount of storage initially because it is more efficient than having to extend the storage. QJOBMSGQSZ is numeric and must be specified in K bytes. The lower limit for QJOBMSGQSZ is 1. QJOBMSGQSZ may be changed to a smaller value by starting OS/400 if the combination of allocation system values requires more auxiliary storage than is available to start the system. If this happens, you will be notified.

QJOBMSGQTL: Maximum initial size of the job message queue. When a job message queue is reinitialized for use by another job, it is truncated to its initial size (the system value QJOBMSGQSZ) if it is larger than the maximum initial size (QJOBMSGQTL). QJOBMSGQTL is numeric and must be specified in K bytes. The lower limit for QJOBMSGQTL is 1. A job message queue is extended, as needed, to the maximum size of a space object (16 megabytes).

QJOBSPLA: Initial size of the spooling control block for a job. (There is one spooling control block for each job in the system.) The spooling control block records information about inline spooled files and output spooled files. This value primarily affects auxiliary storage requirements and has little effect on processing performance. The auxiliary storage is retained for every job known to the system.

The allocated area is made up of standard control information plus a separate set of control information for each spooled file. The value you supply is incremented by approximately 200 bytes and then rounded up to the nearest 512-byte page boundary. The default is 1536 bytes, which rounds up to four pages (2048 bytes or 2K) and allows for about three output spooled files per job.

If your typical job uses more than the three output files and you are not concerned with an additional 2K allocation per job, a good choice would be 3600. This rounds up to 4096 bytes and allows for an initial allocation of approximately seven or eight spooled output files per job. The next largest logical choice would be 7600, which would round up to 8K. Set the QJOBSPLA value so that it rounds up to a 2K, 4K, or 8K value.

If the number of spooled files created by a job exceeds the initial allocation, the value assigned to the QADLSPLA system value is used to extend the control block.

QJOBSPLA is numeric. The lower limit is 1024 bytes. A change to this value takes effect when a cold start is requested during OS/400 installation. If the system requires new job structures, the value is also used for the new job structures.

QTOTJOB: Initial number of jobs for which auxiliary storage is to be allocated during IPL. The number of jobs is the number supported by the system at any one time, which includes the jobs on job queues, active jobs (including system jobs), and jobs having output on output queues. The amount of auxiliary storage allocated for each job is approximately the sum of the QJOBSPLA and QJOBMSGQSZ system values.

Note: The system value QJOBSPLA is measured in bytes and the system value QJOBMSGQSZ is measured in K bytes.

QTOTJOB is numeric. The lower limit for QTOTJOB is 1. A change to this value takes effect at the next IPL unless the change is made on the IBM-supplied configuration menu, in which case, it is used for the current IPL. QTOTJOB may be changed to a smaller value during IPL if the combination of allocation system values requires more auxiliary storage than is available to start the system. If this happens, you will be notified.

To find the number of total jobs in the system, view the third line of the system status display (using the WRKSYSSTS command). This number should usually be kept within reason as it is a factor in the time to perform IPL and some internal searches. In general, it is preferable to keep this value below 300. This may require periodic removal of jobs that have only job logs. The *CL Programmer's Guide* has a discussion of job logs and how to remove them for jobs that complete normally. As long as a job has one or more spooled output files known to the system, knowledge of the job remains in the system and counts toward the display system status value. A reasonable value to assign to QTOTJOB is two to seven times the value used for QACTJOB, depending on how often you clear your output queues.

Message and Logging System Values

QACGLVL: Accounting level. Its value is a 80-character list of accounting options. Apostrophes are necessary only when more than one option is specified and each option must be separated by blanks. The possible accounting level options are:

Accounting Option	Accounting Information
*NONE	No accounting information is journaled.
*JOB	Job resource use is journaled.
*PRINT	Spooled and nonspooled print file resource use is to be journaled.

Note: *NONE cannot be used with the other options.

To change the QACGLVL system value, the user must have security officer (QSECOFR) authority, which can be from a user profile or a group profile or it can adopted by a program. When the QACGLVL system value is changed to options other than *NONE, the system accounting journal QSYS/QACGJRN must exist; if not, the change is rejected. The QACGLVL system value cannot be changed on the IBM-supplied configuration menu during IPL. A change to the QACGLVL system value does not affect jobs already on the system, but does affect new jobs that enter the system after the change. For more information on job accounting, see Chapter 8, "Using Job Accounting" on page 8-1.

QHSTLOGSIZ: Maximum number of records for each version of the history log. When a version is full (the maximum has been reached), a new version is created. You can save the full (old) version on diskettes and then delete it. QHSTLOGSIZ is numeric. The lower limit for QHSTLOGSIZ is 1. (See the *CL Programmer's Guide* for more information about QHST.)

QPRTTXT: Print text. This system value is used to print up to 30 characters of text on the bottom of listings and separator pages. In order for the text to be centered at the bottom of the listing, it must be centered in the QPRTTXT value field. The QPRTTXT system value will be used for system and subsystem monitor jobs and will be the system-wide default print text for all other jobs. If *BLANK is entered, QPRTTXT will be set to all blanks. The *CL Reference* has more information on formatting print text.

QSRVDMP: Provides for user control over service dumps for unmonitored escape messages. The allowable values are:

- *DMPUSRJOB: Request dumps only in user jobs, not in system jobs. (System jobs include the system arbiter, subsystem monitors, LU services process, spool readers and writers, and the SCPF job.)
- *DMPSYSJOB: Request dumps only in system jobs, not user jobs.
- *DMPALLJOB: Request dumps in all jobs.
- *NONE: Do not request dumps in any jobs.

Note: The ability to control the creation of service dumps with this system value may have an effect on IBM's ability to respond to some system failures. The value *DMPALLJOB can be used to ensure that a service dump is created whenever a failing function requests one. The initial value (*DMPUSRJOB) provides that service dumps are created only for user jobs.

Storage System Values

QBASACTLVL: Maximum activity level of the base storage pool. This value indicates how many jobs (system and user jobs) can compete at the same time for storage in the base storage pool. QBASACTLVL depends on the types of jobs being run in this storage pool. A change to this value takes effect when a subsystem having storage pools other than the base storage pool is started, when the storage size or activity level of a storage pool in an active subsystem is changed, or when the subsystem is ended. QBASACTLVL is numeric. The lower limit for QBASACTLVL is 1.

QBASPOOL: Minimum size of the base storage pool. Storage not allocated to another storage pool remains in the base storage pool. A change to this value takes effect immediately. In some circumstances, a machine function may be using storage allocated to the base pool. If this is so, and if the change to this system value would reduce the allocation to less than 32K plus the amount needed by the machine, the system value is changed immediately but the actual base pool change is done by the OS/400 only after the storage in use is released by the machine. Use the Work with System Status (WRKSYSSTS) command to determine the amount of storage reserved for machine functions. QBASPOOL is numeric, must be specified in K bytes, and cannot be set to less than 32K.

QMAXACTLVL: Maximum activity level of the system. This is the number of jobs that can compete at the same time for main storage and processor resources. For all active subsystems, the sum of all jobs running in all storage pools cannot exceed QMAXACTLVL. If a job cannot be processed because the activity level has been reached, the job is held until another job reaches a time slice or a long wait. QMAXACTLVL is numeric. The lower limit for QMAXACTLVL is 2. A change to this value takes effect immediately.

QMCHPOOL: Size of the machine storage pool. The machine storage pool contains highly shared machine and OS/400 programs. You must be careful when changing the size for this storage pool because system performance may be impaired if the storage pool is too small. A change to this value takes effect immediately. QMCHPOOL is numeric, must be specified in K bytes, and cannot be set to less than 256 K bytes.

This minimum value of 256 K bytes, which is enforced by the Change System Value (CHGSYSVAL) command, does not reflect the machine-enforced minimum value. The machine-enforced minimum value varies depending on the main storage size of the machine. The system automatically increases the actual size of the machine storage pool to the machine-enforced minimum value if the QMCHPOOL system value contains a smaller value.

If the system has increased the actual size of the machine storage pool, you can use the Work with System Status (WRKSYSSTS) command to determine the actual machine-enforced minimum value for the machine storage pool (pool 1).

Chapter 5, "Performance Guidelines" on page 5-1 contains a formula for determining the size of the machine storage pool for your system. You should use the result of this calculation, rather than the machine-enforced minimum value, to set the QMCHPOOL system value when you tune the performance of your system.

Network Attributes

Network attributes contain specifications that can be used for networking and communications. Network attributes are not objects and cannot be passed as parameter values like CL variables.

The system is shipped with certain network attributes. Most of these are important only if your system is part of a communications network. However, you may want to change the system name because it appears on many OS/400 displays, including the sign-on display.

The network attributes shipped with the system are shown in the following table. For more information about each network attribute, see the *CL Reference*, as well as the manuals listed in the table.

Description	**Shipped Value**	**Notes**
System name	SYSNAME(based-on-machine-serial-number)	See the *C & SM User's Guide.*
Local network ID to be assigned to the system	LCLNETID(APPN)	See the *Communications User's Guide* and the *APPC and APPN User's Guide.*
Local control point name for the system	LCLCPNAME(based-on-machine-serial-number)	See the *Communications User's Guide* and the *APPC and APPN User's Guide.*
Default local location name for the system	LCLLOCNAME(based-on-machine-serial-number)	See the *Communications User's Guide* and the *APPC and APPN User's Guide.*
Default mode name for the system	DFTMODE(BLANK)	See the *Communications User's Guide* and the *APPC and APPN User's Guide.*
Maximum number of conversations for a remote location	MAXLOCCNV(64)	See the *APPC and APPN User's Guide.*
APPN node type	NODETYPE(*ENDNODE)	See the *APPC and APPN User's Guide.*
Maximum number of intermediate sessions	MAXINTSSN(200)	See the *APPC and APPN User's Guide.*
APPN route addition resistance	RAR(128)	See the *APPC and APPN User's Guide.*
APPN network node servers	NETSERVER	See the *APPC and APPN User's Guide.*
Alert status	ALRSTS(*OFF)	See the *C & SM User's Guide.*

Figure 7-1 (Part 1 of 2). Network Attributes Shipped with the System

Description	**Shipped Value**	**Notes**
Specifies if the system is an alert primary focal point	ALRPRIFP(*NO)	See the *C & SM User's Guide.*
Specifies if the system is an alert default focal point	ALRDFTFP(*NO)	See the *C & SM User's Guide.*
Specifies which alerts will be logged	ALRLOGSTS(*NONE)	See the *C & SM User's Guide.*
Name of the controller through which alerts are sent on the SSCP-PU session	ALRCTLD(*NONE)	See the *C & SM User's Guide.*
Default message queue used in object distribution	MSGQ(QSYS/QSYSOPR)	See the *Distribution Services Network Administrator's Guide.*
Default output queue used in object distribution	OUTQ(QGPL/QPRINT)	See the *Distribution Services Network Administrator's Guide.*
Default action taken for job streams received by the system (used in object distribution)	JOBACN(*FILE)	See the *Distribution Services Network Administrator's Guide.*
Maximum number of systems a distribution can pass through on its path to a destination on a SNADS network	MAXHOP(255)	See the *Distribution Services Network Administrator's Guide.*
The action to be taken for DDM requests from other systems	DDMACC(*OBJAUT)	See the *DDM User's Guide.*
Specifies how PC Support requests will be handled	PCSACC(*OBJAUT)	See the *PC Support User's Guide.*

Figure 7-1 (Part 2 of 2). Network Attributes Shipped with the System

To display the current network attributes, use the Display Network Attributes (DSPNETA) command. To change the network attributes, use the Change Network Attributes (CHGNETA) command. To retrieve network attributes into a CL program, use the Retrieve Network Attributes (RTVNETA) command. To save network attributes with system values, use the Save System (SAVSYS) command. For more information on using these commands, see the *CL Reference.*

Chapter 8. Using Job Accounting

This chapter describes the job accounting function which gathers data so that you can determine who is using your system and what system resources they are using. It also assists you in evaluating the overall use of your system.

Job Accounting

Typical job accounting data details the jobs running in your system and the resources they are using such as the use of the processing unit, printer, display stations, data base and communications functions.

Job accounting is optional. You must take specific steps (described later in the section "Setting Up Job Accounting" on page 8-14) to set up job accounting. You may also assign accounting codes to user profiles or specific jobs.

Job accounting statistics are kept by using the journal entries made in the system accounting journal QSYS/QACGJRN. You should know how to perform journal management operations, such as saving a journal receiver, changing journal receivers, and deleting old journal receivers. For more information about journal management, see the *Backup and Recovery Guide*.

When you want to analyze the job accounting data, it must be extracted from the QACGJRN journal by use of the DSPJRN command. With this command you can write the entries into a data base file. You must write application programs or use a utility such as the query utility to analyze the data.

You may request the system to gather job resource accounting, printer file accounting data, or both. The system provides different journal entries for each type:

- Job resource accounting: The job (JB) journal entry contains data summarizing the resources used for a job or for different accounting codes used in a job.
- Printer file accounting:
  - Direct print (DP) journal entry: Contains data about printer files produced on print devices (nonspooled).
  - Spooled print (SP) journal entry: Contains data about printer files made by a print writer (spooled).

Resource Accounting

Resource accounting data is summarized in the JB journal entry at the completion of a job. In addition, the system creates a JB journal entry summarizing the resources used each time a Change Accounting Code (CHGACGCDE) command occurs. The JB journal entry includes:

- Fully qualified job name
- Accounting code for the accounting segment just ended
- Processing unit time
- Number of routing steps
- Date and time the job entered the system
- Date and time the job started
- Total transaction time (includes service time, ineligible time, and active time)
- Number of transactions for all interactive jobs
- Auxiliary I/O operations
- Job type
- Job completion code
- Number of printer lines, pages, and files created if spooled or printed directly
- Number of data base file reads, writes, updates, and deletes
- Number of OS/400-ICF file read and write operations

Some of the job accounting information can also be accessed using the CPF1124 and CPF1164 messages located in the QHST log. Job accounting data available through the QHST messages is a subset of the method described here. For more information on QHST messages, see the *CL Programmer's Guide*. See "Deciding Whether to Use Job Accounting" on page 8-6 to determine the method to use.

Printer File Accounting

There are two types of journal entries for printer file accounting:

- DP for nonspooled printer files
- SP for spooled printer files

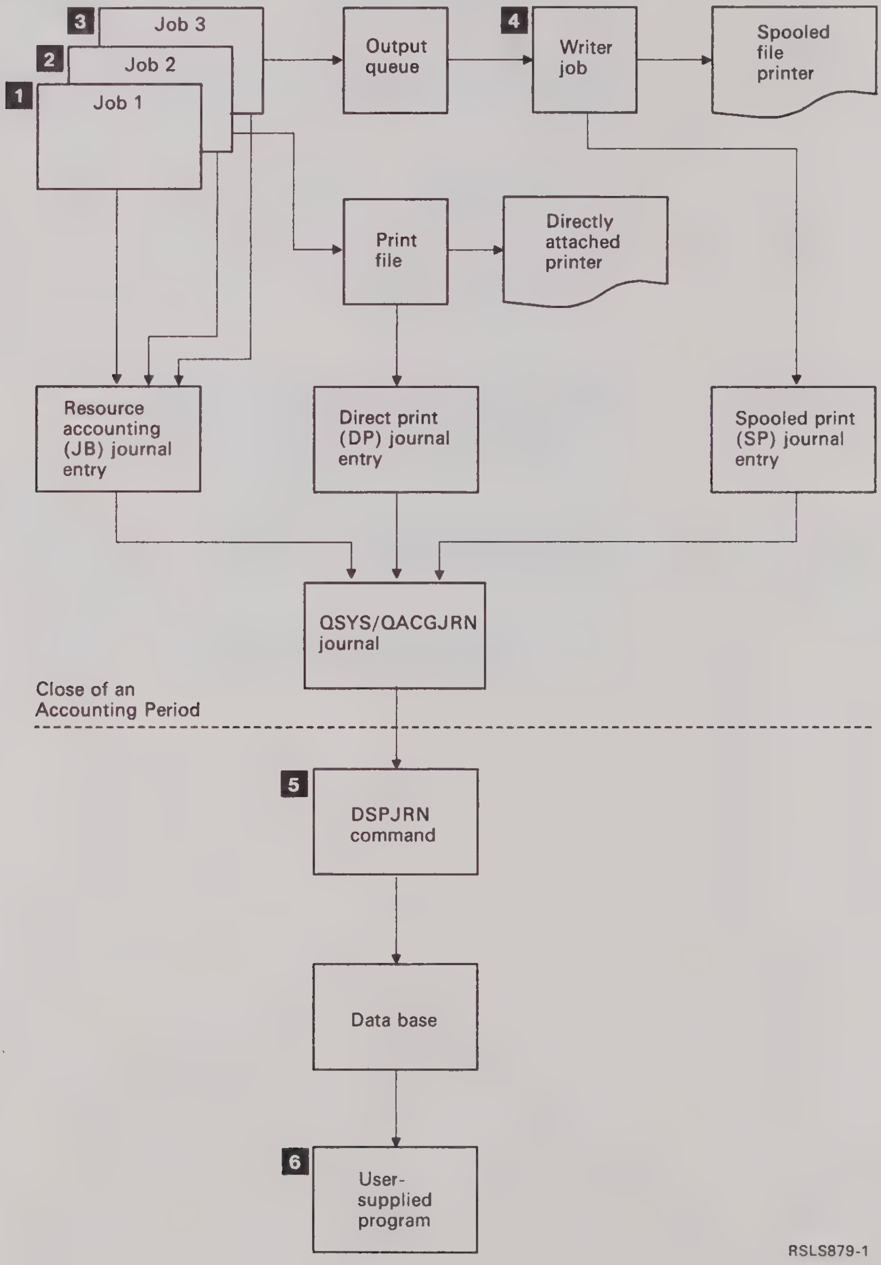
These two types of journal entries share a common journal entry format although some of the information is only available in the SP entry. The DP and SP journal entries include information such as:

- Fully qualified job name
- Accounting code
- Device file name and library
- Device name
- Device type and model
- Total number of pages and lines printed. If multiple copies occurred, this is the sum of all copies.
- Spooled file name (only in the SP entry)
- Spooled file number (only in the SP entry)
- Output priority (only in the SP entry)
- Form type (only in the SP entry)
- Total number of bytes of control information and print data sent to the printer device. If multiple copies occurred, this is the sum of all copies. (This only applies to the SP entry.)

The DP and SP journal entries occur when the file is printed. If a spooled file is never printed, no SP journal entry will appear.

Overview of Job Accounting

The following figure provides an overview of job accounting as provided on the system:


3
Job 3
2
Job 2
1
Job 1
Output queue
4
Writer job
Spooled file printer
Print file
Directly attached printer
Resource accounting (JB) journal entry
Direct print (DP) journal entry
Spooled print (SP) journal entry
QSYS/QACGJRN journal
Close of an Accounting Period
5
DSPJRN command
Data base
6
User-supplied program
RSLS879-1

1 When job 1 is completed, the system summarizes the resources used and writes the JB journal entry to the QACGJRN journal. If the accounting code was changed during the job, a JB journal entry would be written for each time the accounting code was changed and at the end of the job. Job 1 does not make any printer output, and no job log is made. Therefore, no DP or SP journal entries are made for job 1.

2 Job 2 is printing a file directly to a printer. When the file is completed, a DP journal entry is written summarizing the printed data. When job 2 is completed, the system summarizes the resources used and writes the JB journal entry. Job 2 does not make any spooled printer output and no job log is made. Therefore, no SP journal entry is made for job 2.

3 Job 3 is printing to a file that is spooled. The SP journal entry is not written unless a print writer prints the file. When job 3 is completed, the system summarizes the resources used and writes the JB journal entry. If a job log is made at the completion of the job, it is a normal spooled file and an SP journal entry is created if the file is printed.

4 A print writer is started to print the files made by one or more jobs. When the writer finishes a file, it makes an SP journal entry. The SP journal entry is not made if the file is canceled before printing starts.

5 At the close of an accounting period (or whenever desired), the DSPJRN command can be used to write the accumulated journal entries into a data base file.

6 User-written programs or the query utility can be used to analyze the accounting data. Reports such as resources used would compile data by a specific:

- Accounting code
- User
- Job type

Job Accounting Operating Characteristics

The AS/400 system attempts to allocate main storage as efficiently as possible. A job may not use the same amount of resources each time it is run. For example, if there are several active jobs on your system, a job spends more time reestablishing the resources needed for running than if a dedicated system environment is used. The system uses the job and run priorities assigned to different jobs to assist in managing main storage. Therefore, high priority jobs use less system resource than low priority jobs.

Because of these system operating characteristics, you may want to apply your own interpretation or algorithm to the job accounting data collected. If you are charging for the use of your system, you may want to charge more for:

- High priority jobs
- Work done during peak system time
- Use of critical resources

Deciding Whether to Use Job Accounting

Because the QHST messages (CPF1124 and CPF1164) are always available in the QHST log, the first question to answer is: Which method should I use to gather job accounting statistics? The following guidelines should help you answer the question.

Job accounting has all the information supplied by CPF1164 plus:

- Accounting code
- Number of print files, lines, and pages created by programs
- Number of data base read, write, and update operations
- Number of communications read and write operations
- Actual lines and pages printed
- Time the job was active and suspended
- Actual number of bytes of control information and print data sent to the printer

The job accounting function is more effective for gathering job accounting statistics if:

- The resource information regarding data base, printer, and communications use is important.
- Accounting codes are assigned to users or jobs.
- The information for printed output is important.
- Job accounting must be done on an accounting segment basis in a job rather than on a complete job basis.
- The active and suspended time information is needed.

The QHST messages are more effective for gathering job accounting statistics if:

- You do not want to manage the additional objects included in journaling.
- You do not need any resource information other than that provided in the CPF1124 and CPF1164 messages, which are sent automatically to the QHST log.
- You do not need print accounting information.

Some statistics recorded in the CPF1164 message and the JB journal entries will not match exactly. This is due mainly to two factors: CPF1164 statistics are recorded slightly before the JB journal statistics; and each time an accounting code is changed, rounding occurs for some fields, while rounding occurs only once for CPF1164 messages.

Accounting Codes

The initial accounting code (up to 15 characters in length) for a job is determined by the value of the ACGCDE (accounting code) parameter in the job description and user profile for the job. When a job is started, a job description is assigned to the job. The job description object contains a value for the ACGCDE parameter. If the default of *USRPRF is used, the accounting code in the job's user profile is used. Note, however, that when a job is started using the Submit Job (SBMJOB) command, its accounting code is the same as that of the submitter's job. You can change the accounting code after the job has entered the system by using the Change Accounting Code (CHGACGCDE) command.

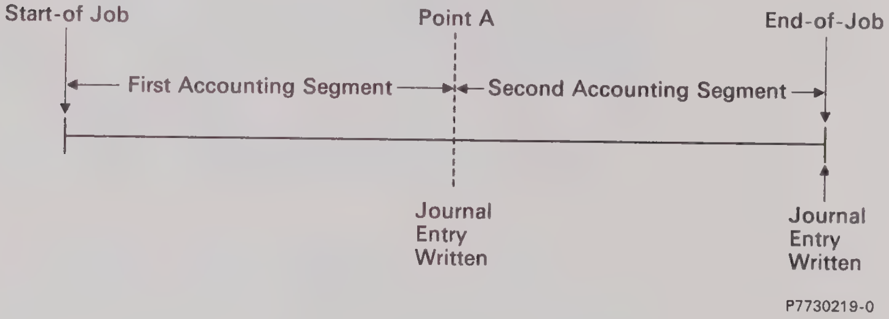
The CRTUSRPRF and CHGUSRPRF commands support the ACGCDE parameter. The default is *BLANK. If all work for a particular user is to be recorded under one accounting code, only user profiles need to be changed. You can change the accounting codes for specific job descriptions by specifying the desired accounting code for the ACGCDE parameter on the CRTJOBD and CHGJOBD commands. The CHGACGCDE command also allows different accounting codes in a single job. You can develop a method of verifying the assignment of accounting codes as described in “Validity Checking of Accounting Codes” on page 8-20 later in this chapter.

The Retrieve Job Attributes (RTVJOBA) command allows you to access the current accounting code in a CL program.

Resource Accounting Data

A JB journal entry is written at the end of every job and any time the job accounting code is changed by the CHGACGCDE command. A JB journal entry is written even if the accounting code is changed while the job is on the job queue although no resources have been used. Each resource accounting journal entry contains information about the resources used while the previous accounting code was in effect.

For example, the following illustrates a job with two accounting segments:



At point A, the CHGACGCDE command was issued. The accounting code is changed and the JB journal entry is sent to the journal. The JB journal entry contains data for the first accounting segment. When the job ends, a second JB entry is made for the job containing data for the second accounting segment.

If the job accounting code was not changed during the existence of the job, the single JB entry summarizes the total resources used by the job. If the job accounting code was changed during the existence of the job, then you must add up the fields in the multiple JB entries in order to determine the total resources used by the job. The creation of a job log does not count toward the processing unit use for a job or its printed output in the JB accounting entries. However, if you are using print file accounting, the job log printed will be included in the printer file journal entries.

General Accounting Journal Information

Each journal entry contains the standard prefix fields for any journal entry (for example, date, time, journal sequence number). See the *Backup and Recovery Guide* for a discussion of these fields.

JB Accounting Journal Information

The following lists the various fields (found in field reference file QSYS/QAJBACG) in the JB journal entry:

Field Name	Description	Field Attributes	Comments
JAJOB	Job name	Character (10)	
JAUSER	Job user	Character (10)	
JANBR	Job number	Zoned (6,0)	
JACDE	Accounting code	Character (15)	
JACPU	Processing unit time used (in milliseconds)	Packed decimal (11,0)	See Note 1
JARTGS	Number of routing steps	Packed decimal (5,0)	
	Job entered the system		
JAEDTE	– Job entry date (mmddyy format)	Character (6)	
JAETIM	– Job entry time (hhmmss format)	Character (6)	
	Job start date and time		See Note 2
JASDTE	– Job start date (mmddyy format)	Character (6)	
JASTIM	– Job start time (hhmmss format)	Character (6)	
JATRNT	Total transaction time (in seconds)	Packed decimal (11,0)	
JATRNS	Number of transactions	Packed decimal (11,0)	See Note 3
JAAUX	Auxiliary I/O operations and data base operations (including page faults for any reason)	Packed decimal (11,0)	
JATYPE	Job type	Character (1)	See Note 4
JCCDE	Completion code	Packed decimal (3,0)	See Note 5
JALINE	Number of print lines	Packed decimal (11,0)	See Note 6
JAPAGE	Number of printed pages	Packed decimal (11,0)	
JAPRTF	Number of print files	Packed decimal (11,0)	
JADBPT	Number of data base write operations	Packed decimal (11,0)	See Note 7
JADBGT	Number of data base read operations	Packed decimal (11,0)	See Note 7
JADBUP	Number of data base update, delete, FEOD, release, commit, and rollback operations	Packed decimal (11,0)	See Note 7
JACMPT	Number of communications write operations	Packed decimal (11,0)	See Note 8
JACMGT	Number of communications read operations	Packed decimal (11,0)	See Note 8
JAACT	Time job was active (in milliseconds)	Packed decimal (11,0)	

Field Name	Description	Field Attributes	Comments
JASPN	Time job was suspended (in milliseconds)	Packed decimal (11,0)	

Notes:

1. The processing unit time does not include processing unit use and printer statistics for the creation of job logs.
2. For job completion date and time from journal entries, use the JODATE and JOTIME fields that are part of the standard journal entry prefix information. (See the *Backup and Recovery Guide*, for more information on these fields.) After an abnormal system ending, these fields contain the current date and time and not (as with CPF1164 messages) the actual time of the system ending.
3. The last transaction (SIGNOFF) is not counted.
4. The job types recorded are the following:

A	Autostart job
B	Batch job (includes communications and MRT)
I	Interactive job
M	Subsystem monitor
R	Spooling reader
W	Spooling writer

Note: These are the same as those used in message CPF1164, except that message CPF1164 includes some system job information not included in the journal entries.

5. The completion codes, which are similar to those used for message CPF1164, are:

000	Normal completion
010	Normal completion during controlled end or controlled subsystem end
020	Job exceeded end severity
030	Job ended abnormally
040	Job ended before becoming active
050	Job ended while active
060	Subsystem ended abnormally while job was active
070	System ended abnormally while job was active
080	Job completed in the time limit
090	Job forced to complete after the time limit has ended
099	Accounting entry caused by CHGACGCDE command

6. The number of print lines does *not* reflect what is actually printed. Spooled files can be canceled or printed with multiple copies. The information in the JB journal entry reflects only what was written by the program. This excludes any lines written for the job log. See the discussion on DP and SP printer file accounting data later in this chapter.
7. The numbers recorded for data base I/O operations do not include I/O operations to readers and writers, or I/O operations caused by the CL commands CPYSPLF, DSPSPLF, or WRKSPLF. If SEQONLY(*YES) is in effect, these numbers show each block of records read, not the number of individual records read.
8. The numbers recorded for communications I/O operations do not include remote work station activity. When the I/O is for a communications device, the numbers include only activity related to OS/400-ICF files.

DP and SP Printer File Accounting Data

The accounting code used for the DP or SP journal entries is the accounting code of the job at the time the file is closed. Sometimes a DP or SP entry is created before the file is closed (such as when a writer which is creating a SCHEDULE(*IMMED) file is ended). When this happens the current accounting code of the job is used.

A DP or an SP journal entry is created for each file printed. If the job log is spooled and then printed, an SP entry is created for it. Also, an SP entry is written for diskette spooled files redirected to a printer by the print writer.

DP Accounting Journal Information

The following lists the various fields (found in file QSYS/QAPTACG) in the DP journal entry:

Field Name	Description	Field Attributes
JAJOB	Job name	Character (10)
JAUSER	Job user	Character (10)
JANBR	Job number	Zoned (6,0)
JACDE	Accounting code	Character (15)
JADFN	Device file name	Character (10)
JADFNL	Library in which device file is stored	Character (10)
JADEVN	Device name	Character (10)
JADEVT	Device type	Character (4)
JADEVM	Device model	Character (4)
JATPAG	Total number of print pages produced	Packed decimal (11,0)
JATLIN	Total number of print lines produced	Packed decimal (11,0)
JASPFN	Always blank	Character (10)
JASPNB	Always blank	Character (4)
JAOPTY	Always blank	Character (1)
JAFMTP	Always blank	Character (10)
JABYTE	Always zero	Packed decimal (15,0)

SP Accounting Journal Information

The information provided is similar to that provided in the DP accounting journal data except that the spooled file name, spooled file number, output priority, form type, and total number of bytes of control information and print data sent to the printer are included.

An SP journal entry is not written if a spooled file is deleted before a writer starts writing the file to the device.

The following lists the various fields (found in file QSYS/QAPTACG) in the SP journal entry:

Field Name	Description	Field Attributes	Comments
JAJOB	Job name	Character (10)	
JAUSER	Job user	Character (10)	
JANBR	Job number	Zoned (6,0)	
JACDE	Accounting code	Character (15)	
JADFN	Device file name	Character (10)	
JADFNL	Library in which device file is stored	Character (10)	
JADEVN	Device name	Character (10)	
JADEVT	Device type	Character (4)	
JADEVM	Device model	Character (4)	
JATPAG	Total number of print pages produced	Packed decimal (11,0)	See Notes 1, 2, 3, and 4
JATLIN	Total number of print lines produced	Packed decimal (11,0)	See Notes 1, 2, 3, and 4
JASPFN	Spooled file name	Character (10)	
JASPNB	Spooled file number	Character (4)	
JAOPTY	Output priority	Character (1)	
JAFMTP	Form type	Character (10)	
JABYTE	Total number of bytes sent to the printer	Packed decimal (15,0)	See Notes 1, 2, 3, and 4

Notes:

1. The system attempts to record the actual number of pages, lines, and bytes printed, but when a writer is canceled *IMMED or recovers from a device error (such as end of forms), it is not possible to determine the exact number of pages, lines, and bytes printed.
2. Extra pages and lines produced with the alignment line are not included in the page, line, and byte counts.
3. The page, line, and byte counts for the job and file separators are included with the counts for the file they are associated with.
4. When an IPDS file contains graphics or bar codes and is sent to an IPDS printer that does not support graphics or bar codes, the page, line, and byte counts include the unprinted graphics and bar codes.

Batch Processing

Any batch job submitted during the running of a job using the SBMJOB command automatically uses the same accounting code as the job that submitted the batch job. When the SBMJOB command is used, the accounting codes cannot be overridden regardless of how the job description entry is coded. If you want the batch job to operate under an accounting code other than that of the submitting jobs, a CHGACGCDE command should be issued either:

- Before and after the SBMJOB command is issued
- Immediately by the batch job

Batch jobs submitted using a reader or a SBMDBJOB or SBMDKTJOB command use the accounting code specified in the job description for the batch job. If the job description specifies ACGCDE(*USRPRF), the accounting code is taken from the user profile used for the job.

Interactive Processing

If an interactive job has a fixed set of options for a user and each option has an assigned accounting code, it may be desirable to automatically assign a new code when the user requests to work on a new function. A typical approach would be for a menu option to request a new functional area. The CHGACGCDE command would then be issued within a CL program and the job values used for the previous accounting code would be summarized in the JB accounting journal entry.

If a user has several assignments for which only he knows the accounting codes to be used, you can:

- Give authority to the user to enter the CHGACGCDE command.
- Write a program to prompt the user for the accounting code.

System Job Processing

System jobs that you control (for example, readers and writers) are assigned an accounting code of *SYS. Other system jobs that you do not control (for example, QSYSARB, QLUS, SCPF) do not receive a journal entry.

Note: You cannot use the CHGACGCDE command to change the accounting code of the subsystem monitor or a reader or writer. You can, however, change the accounting code of a reader or writer by changing the appropriate IBM-supplied job descriptions and user profiles and then starting them again.

Setting Up Job Accounting

To set up resource or printer file accounting, perform the following:

1. Create a journal receiver in a library of your choice by using the Create Journal Receiver (CRTJRNRCV) command:

   ```
   CRTJRNRCV  JRNRCV(USERLIB/ACGJRN1)
   ```

   You should name the journal receiver ACGJRN1 or a similar name that can be used to create a naming convention such as ACGJRN2, ACGJRN3, for future journal receivers.

   After you create the first receiver, you can create additional receivers and attach them to the QSYS/QACGJRN journal automatically with the correct naming convention by using the CHGJRN JRNRCV(*GEN) command. You will probably want to place the journal receiver in one of your libraries that is saved regularly.

   You can specify options on the CRTJRNRCV command to assist in your operational aspects. If you want to use dual journal receivers, you must create a second journal receiver.

2. Create the journal QSYS/QACGJRN by using the Create Journal (CRTJRN) command. The name QSYS/QACGJRN *must* be used, and you must have authority to add objects to QSYS. You need to specify the name of the journal receiver(s) you created in the previous step and any other options on the command. You should also consider who you want to have authority to this journal.

3. As the security officer, change the accounting level system value QACGLVL using the CHGSYSVAL command.

   The VALUE parameter on the CHGSYSVAL command determines when job accounting journal entries are produced:

VALUE Parameter	Description
*NONE	The system does not produce any entries in the job accounting journal.
*JOB	The system produces a JB journal entry for each accounting segment of a job.
*PRINT	The system produces a DP or an SP journal entry for each file printed (either a nonspooled file or a spooled file written by a print writer).
*JOB *PRINT	The system produces both types of journal entries.

   The system requires that the QSYS/QACGJRN journal be created before this system value is changed to request job accounting.

For information on processing the accounting journal entries, see the section "Converting Job Accounting Journal Entries" on page 8-19 later in this chapter.

Accounting Journal

The accounting journal QSYS/QACGJRN is processed as any other journal. Files can also be journaled to it although for simplicity it is recommended that you keep it solely for accounting information. You can use the SNDJRNE command to send other entries to this journal. While there are additional operational considerations involved in using several journals, there are advantages to *not* allowing any file entries in the QACGJRN journal. It is usually easier to control the QACGJRN journal separately so that all job accounting entries for a particular accounting period are in a minimal number of journal receivers and that a new journal receiver is started at the beginning of an accounting period. System entries also appear in the journal QACGJRN. These are the entries with a journal code of J, which relate to IPL and general operations performed on journal receivers (for example, a save of the receiver).

Job accounting entries are placed in the journal receiver starting with the next job that enters the system after the CHGSYSVAL command takes effect. The accounting level of a job is determined when it enters the system. If the QACGLVL system value is changed after the job is started, it has no effect on the type of accounting being performed for that job. The DP and SP print entries occur if the job that created the file is operating under accounting and the system value is set for *PRINT.

If damage occurs to the journal or to its current receiver so that the accounting entries cannot be journaled, a CPF1302 message is sent to the QSYSOPR message queue, and the accounting data is written to the QHST log in the CPF1303 message. The job trying to send the journal entry continues normally. Recovery from a damaged journal or journal receiver is the same as for other journals. See the *Backup and Recovery Guide* for recovery considerations.

The journal QACGJRN should not be allocated by another job. If the journal is allocated by another job, the journal entry is changed to message text and sent to the QHST log as message CPF1303.

You can use the OUTFILE parameter on the Display Journal (DSPJRN) command to write the accounting journal entries to a data base file that you can process.

You can also use the RCVJRNE command on the QACGJRN journal to receive the entries as they are written to the QACGJRN journal.

Analyzing Job Accounting Data

While the most obvious use for the accounting data is to charge users for system resources, you may want to analyze the data in other ways. Some methods of analyzing the job accounting information would be sequencing by the following fields or combinations of fields:

- Accounting code
- User
- Job type (for example, batch or interactive)
- Time of day

Each of these sequences could be processed by a separate logical file.

A typical statistical analysis of a job would be a summary which reports on the:

- Type of job run with a count of occurrences and totals on processing unit time used
- Interactions
- Average response time
- Auxiliary I/O and file counts by type of file

Other items you may want to analyze are:

- Job termination code
- User profile name
- All jobs that exceed a certain amount of processing unit time

Security Considerations

Only the security officer (or a program adopting his authority) or a user with *ALLOBJ special authority can change the system value QACGLVL. The change takes effect when a new job enters the system. This restriction ensures that if job accounting is in effect and the security officer performs system IPL, an accounting entry is written for the security officer's job.

You can assign job accounting codes only if you have the authority to use the CRTUSRPRF, CHGUSRPRF or CHGACGCDE command. This restricts the use of accounting codes and provides a basis for validity checking any changes.

Only a user with the *SECADM special authority is allowed to use the CRTUSRPRF and CHGUSRPRF commands. However, the security officer can delegate this authority by creating a CL program, which allows another user to adopt the security officer's profile and change the ACGCDE parameter in the user profile. The CL program could then have authority to one or more individuals.

The ACGCDE parameter also exists in job description objects, but the user must have the authority to use the CHGACGCDE command to enter a value other than the default of *USRPRF. This command is supplied with *CHANGE authority.

If you allow a user to use the CHGACGCDE command, he can:

- Create or change the ACGCDE parameter in job descriptions. (Authority to create or change job descriptions is also required.)
- Change the accounting code in his current job.

- Change the accounting code of a job other than his own if he also has the *JOBCTL special authority.

You can provide additional security by using the CHGACGCDE command in a CL program, which adopts the program owner's authority. This allows the user who is running an external function to perform a security-sensitive function without having direct authorization to the CHGACGCDE command.

The accounting journal and its receivers are treated as any other journal objects from a security viewpoint. Refer to the *Backup and Recovery Guide*, for a discussion of the journal security considerations. You must decide what authorization should exist for the accounting journal and journal receiver.

Recovery Considerations

If a job ends abnormally, the final accounting entry is written and all previously written accounting entries appear in the journal.

If an abnormal system ending occurs, the following accounting data is lost to the last routing step or last end-of-accounting segment, whichever occurred most recently.

- Information on the number of lines and pages printed
- Number of files created
- Data base put, get, and update operations
- Communications put and get operations
- Auxiliary I/O operations
- Transaction time
- Number of transaction fields
- Time active
- Time suspended

Transaction time may be logged with a negative 1 (-1). This happens when the system date is moved ahead while an interactive job is in the middle of a transaction.

After an abnormal system ending, the job completion time in the journal will not be the same as that in the CPF1164 message. The message uses the time nearest to that of the system ending, but the job accounting journal entries are sent to the journal during IPL, and the job completion time is the current system time, which is later than the time when the abnormal system ending occurred.

If the system ends abnormally, some journal entries can be lost. These are the entries that are written to the journal but not forced to disk (this is equal to FORCE(*NO) on the SNDJRNE command). They include the following:

- JB entries caused by a CHGACGCDE command
- DP and SP entries

Whenever a job completes, the last accounting code entry is forced to disk (as if FORCE(*YES) were specified on the SNDJRNE command). Whenever an accounting entry is forced to disk, all earlier entries in the journal, regardless of which job produced them, are forced to disk.

Exception: If only *PRINT accounting is specified on the system, there will not be any job ending FORCE(*YES) journal entries done. Therefore, if a critical accounting entry is written by a CHGACGCDE command and you want to ensure it will not be lost in case of an abnormal system ending, you could issue a SNDJRNE command and specify the FORCE(*YES) option. If files are also to be journaled to the accounting journal, any data base changes are always forced to the journal, and this causes all earlier accounting entries to also be forced.

If an abnormal system ending occurs or you change an accounting code of a job other than your own, the qualified job name in the first 30 bytes of the JARES field in the journal entry describe the system job that wrote the JB entry at the next IPL and not the job that used the resources. The JAJOB, JAUSER, and JANBR fields should be used for analysis purposes.

If the job accounting journal or journal receivers become damaged, the system continues to operate and to record accounting data in the history log. To recover from the journal or journal receiver damage, use the Work with Journal (WRKJRN) command. Refer to the *Backup and Recovery Guide* for more information about the use of this command. After recovering the damaged journal or journal receiver, change the system value QACGLVL to a value appropriate for your installation. (Unless you change the QACGLVL system value, the system will not record accounting information in the new journal receiver.)

If you need to access the information from the CPF1303 message you will need to create a high-level language program. In order to define records that match the CPF1303 message you will need to include the following fields:

System Time	Char (8)
Message Record Number	Bin (4)
Qualified Job Name	Char (26)
Entry Type (JB, DP, or SP)	Char (2)
Length of Data	Bin (2)

Followed by fields:

JAJOB through JASPN for JB entries
JAJOB through JABYTE for SP and DP entries

(These fields are described earlier in this chapter.)

For an example program, refer to the section in *CL Programmer's Guide*, that discusses processing the QHST file for the job completion message.

The CPF1164 message always consists of three records and the CPF1303 message always consists of four records.

The information contained in the standard journal prefix fields is not included in this message. All that is needed is information pertaining to the job end, date, and time; this information can be found in record 1 of the CPF1303 message.

Converting Job Accounting Journal Entries

You can use the OUTFILE parameter on the DSPJRN command to write the job accounting journal entries into a data base file that you can process. The OUTFILE parameter allows you to name a file or member. If the member exists, it is cleared before the records are written. If the member does not exist, it is added. If the file does not exist, a file is created using the record format QJORDJE. This format defines the standard heading fields for each journal entry, but the job accounting data is defined as a single large field.

To avoid having to process the accounting data as a single large field, two field reference files are supplied to help you in processing the job accounting journal entries. The file QSYS/QAJBACG contains the record format QWTJAJBE and is used for JB entries. File QSYS/QAPTACG contains record format QSPJAPTE and is used for DP or SP entries. The same format is used for all printer file entries regardless of if the output is SP (spooled) or DP (nonspooled). The DP entry for directly printed files contains some fields that are not used; these fields contain blanks.

The following are some approaches you might use:

- The basic JB entries and DP or SP entries can be processed by creating two output files using the supplied field reference file formats and running the DSPJRN command once for JB and once for DP or SP. This allows you to define a logical file over the two physical files and use an high-level language program to process the externally described file. This solution is recommended and described later in this section.
- You can process only the JB entries by creating a file using one of the supplied field reference files (QSYS/QAJBACG) to create an externally described file. This file can then be processed by the query utility or an high-level language program.
- You can convert both types of journal entries using the default DSPJRN format of QJORDJE. You would then use a program-described file to process the journal entries in a high-level language program.

The following DDS defines a physical file for the JB journal entries using the QAJBACG field reference file in QSYS. You would normally create the file (using the CRTPF command) with the same name (QAJBACG) as the model file.

```
R QWTJAJBE   FORMAT(QSYS/QAJBACG)
```

The following DDS defines a physical file for the DP or SP journal entries using the the QAPTACG field reference file in QSYS. You would normally create the file (using the CRTPF command) with the same name (QAPTACG) as the model file.

```
R QSPJAPTE   FORMAT(QSYS/QAPTACG)
```

You can specify a key field in either physical file; however, in this example, a logical file is used for sequencing.

If you create two physical files (one for JB and one for DP or SP) with the members of the same name, you would issue the following DSPJRN commands to convert the entries. Assume that you have created the physical files with the same names as the model files in your library YYYY.

```
DSPJRN  JRN(QACGJRN) JRNCDE(A) ENTTYP(JB) OUTPUT(*NONE) +
        OUTFILE(YYYY/QAJBACG)
DSPJRN  JRN(QACGJRN) JRNCDE(A) ENTTYP(SP DP) OUTPUT(*NONE) +
        OUTFILE(YYYY/QAPTACG)
```

You can control the use and selection criteria of the DSPJRN command so that you do not convert the same entries several times. For example, you can select all entries in a specific range of dates. You could convert all of the entries at a cutoff point for your job accounting analysis, for example, monthly. One or more journal receivers may have been used during the month. Note that each use of the DSPJRN command to the same member causes the member to be cleared before any new entries are added. Do not use the JOB parameter of the DSPJRN command as some entries are made for a job by a system job and will therefore not appear as you may expect them to.

Enter the following DDS to create a logical file to allow processing of both physical files. This allows you to read a single file in accounting code order and print a report using a high-level language program:

```
R QWTJAJBE   PFILE(YYYY/QAJBACG)
K JACDE
R QSPJAPTE   PFILE(YYYY/QAPTACG)
K JACDE
```

If you want to use a logical file to process only the basic job accounting record in accounting code order by a user name, you would enter the following DDS for a logical file:

```
R QWTJAJBE   PFILE(YYYY/QAJBACG)
K JACDE
K JAUSER
```

This logical file can be processed by the query utility or by a high-level language program.

If an abnormal system ending occurs, the qualifed job name in the first 30 bytes of the JARES field in the journal entry describe the system job that wrote the entry at the next IPL and not the job that used the resources. For this reason, any analysis done on the JB entries should use the JAJOB, JAUSER, and JANBR fields.

Job Accounting Approaches

The following approaches may be useful to you in the setting up and managing your job accounting function.

Validity Checking of Accounting Codes

An important aspect of any data processing application is ensuring that the correct control fields are specified. For job accounting codes, this can require a complex validity-checking function which not only checks for the existence of authentic codes, but also checks which users are allowed to use specific codes.

Accounting codes can be assigned in the following areas:

- User profile
- Job description
- In a job (CHGACGCDE command)

Controlling the Assignment of Accounting Codes in User Profiles

If it is important to control the assignment of accounting codes, you may want to consider the following approach. Before an accounting code is placed in a user profile, you may want to ensure that the code is valid and that it is valid for a particular user.

You could control the changing of accounting codes on the CHGJOBD command by giving only the security officer authority to the CHGACGCDE command, and therefore, another user could not create or change a job description with a specific accounting code.

The CHGACGCDE command could be used directly to allow users to change the job accounting code of their own or another job. To change another job, the user must also have the special authorization of *JOBCTL.

In many installations, it would not be valid to allow the change of an accounting code for a job on the job queue or for one job to change the accounting code of another job. Controlling this and ensuring validity checking of the new accounting code could be done with a CL program and command.

For example, the CHGACGCDE command would be privately authorized and included in a CL program where it only changed the current job (for example, JOB(*) is specified). The command would be authorized appropriately.

Glossary

access. To read; the ability to use or read.

access method. A method used to read a record from, or to write a record into a file. Access can be sequential (records are referred to one after another in the order in which they appear in the file), it can be random (the individual records can be referred to in any order), or it can be dynamic (records can be accessed sequentially or randomly, depending on the specific request).

access path. The order that records in a data base file are organized for processing by a program. See *arrival sequence access path* and *keyed sequence access path*.

accounting code. A 15-character field, assigned to a job by the system when it is processed by the system, that is used to collect statistics for the system resources used for that job when job accounting is active.

accounting entry. A journal entry that contains statistics of system resources used for job accounting.

accounting level. A system value identifying the type of data to be journaled when job accounting is active.

accounting segment. The period of time during which statistics are gathered, beginning when the job starts or when the job's accounting code is changed, and ending when the job ends or when the job's accounting code is next changed.

acknowledgment (ACK) character. (1) (BSC) The BSC transmission control character that is sent as a positive response to a data transmission. See also *ACK0* and *ACK1*. (2) (RJE) A transmission control character *sequence* that is sent as a positive response to a data transmission.

acknowledgment. Positive response to a data transmission.

ACK0. The even-numbered, positive acknowledgment character, which indicates that text was received without transmission errors. See *acknowledgment (ACK) character*.

ACK1. The odd-numbered, positive acknowledgment character, which indicates that text was received without transmission errors. See *acknowledgment (ACK) character*.

acquire-program-device operation. An operation that makes a program device available for input or output operations. Contrast with *release-program-device operation*.

active group job. A group job that was not suspended by the Transfer to Group Job (TFRGRPJOB) command.

activity level. A characteristic of main storage or of the processing unit that specifies the maximum number of jobs that can run at the same time in the main storage or in the processing unit.

actuator. (1) The device within an auxiliary storage device that moves the read/write heads. (2) A device that causes mechanical motion.

add authority. A data authority that allows the user to add entries to an object; for example, add job entries to a job queue or add records to a file. Contrast with *delete authority*. See also *read authority* and *update authority*.

address. (1) The location in the storage of a computer where particular data is stored. Also, the numbers that identify such a location. (2) In data communications, the unique code assigned to the location of each device or system connected in a network. (3) The second part of a two-part user identification used to send distributions. See also *user ID/address*.

adopted authority. Authority given to the user by the program for the duration of the job that uses that program. The program must be created with owner authority.

advanced peer-to-peer networking (APPN). Data communications support that routes data in a network between two or more APPC systems that are not directly attached.

advanced program-to-program communications (APPC). Data communications support that allows programs on an AS/400 system to communicate with programs on other systems having compatible communications support. APPC is the AS/400 method of using the SNA LU session type 6.2 protocol.

alert. A record sent to a focal point to identify a problem or an impending problem.

alert controller. The host system, specified on the Change Network Attributes (CHGNETA) command by the ALRCTLD parameter, that collects and processes the alerts.

alert focal point. The location in a network specified as the forwarding node to the host system. An alert focal point is a subset of a problem management focal point.

all authority. An object authority that allows the user to perform all operations on the object except those limited to the owner or controlled by authorization list management authority. The user can control the object's existence, specify the security for the object, and change the object. Contrast with *exclude authority*.

all object authority. A special authority that allows users to use all system resources without having specific authority to the resources. See also *save system authority*, *job control authority*, *security administrator authority*, *service authority*, and *spool control authority*.

allocate. To reserve (lock) a resource for use in performing a specific task. Contrast with *deallocate*.

alternative console. A display device assigned by the operating system to function as the console if the console is not working.

APPC. See *advanced program-to-program communications (APPC)*.

application program. A program used to perform a particular data processing task such as inventory control or payroll.

application program interface. A functional interface supplied by the operating system or a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

APPN. See *advanced peer-to-peer networking (APPN)*.

arrival sequence access path. An access path to a data base file that is arranged according to the order in which records are stored in the physical file. See also *keyed sequence access path* and *access path*.

ASP. See *auxiliary storage pool*.

assumed value. A value supplied by the system when no value is specified by the user.

asynchronous. (1) Not occurring in a regular or predictable pattern. (2) Without regular time relationship.

asynchronous I/O. A series of input/output operations that are being done separately from the job that requested them.

asynchronous processing. A series of operations that are done separately from the job in which they were requested; for example, submitting a batch job from an interactive job at a work station. Contrast with *synchronous processing*.

Attention-key-handling program. A user-defined program that is called when the work station user presses the Attention (Attn) key.

attribute. A characteristic or property of one or more objects.

authorize. Permit or give authority to.

automatically started job entry. A work entry in a subsystem description that specifies a job to be automatically started each time the subsystem is started.

auxiliary storage. All addressable storage other than main storage.

auxiliary storage pool. A group of disk units defined from the auxiliary storage devices. Abbreviated ASP. See also *system ASP* and *user ASP*.

base pool. A storage area that contains all unassigned main storage on the system and whose minimum size is specified in the system value QBASPOOL. The system-recognized identifier is *BASE.

batch. Pertaining to a group of jobs to be run on a computer sequentially with the same program with little or no operator action. Contrast with *interactive*.

batch device. Any device that can read serial input or write serial output, or both, but cannot be used to communicate interactively with the system. Examples of batch devices are printers, magnetic tape units, or diskette units.

batch job. A predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system. Contrast with *interactive job*.

batch subsystem. A part of main storage where batch jobs are processed.

BCC. See *block-check character (BCC)*.

beacon message. A message frame sent repeatedly by an adapter indicating a serious network problem, such as a broken cable. See also *beaconing*.

beaconing. Pertaining to an adapter in a token-ring network that repeatedly sends a frame (beacon message) when it is not receiving a normal signal because of serious error, such as a line break or power failure. The message frame repeats until the error is corrected or bypassed.

binary. (1) * (ISO) Pertaining to a selection, choice, or condition that has two possible values. (2) A numbering system with a base of two (0 and 1).

binary synchronous communications (BSC). A data communications line protocol that uses a standard set of transmission control characters and control character sequences to send binary-coded data over a communications line. Contrast with *synchronous data link control (SDLC)*.

block. (1) A group of records that are recorded or processed as a unit. (2) A set of adjacent records stored as a unit on a disk, diskette, or magnetic tape. (3) In data communications, a group of records that are received, processed, or sent as a unit. (4) A sequential group of statements (defined using line commands) that are processed as a unit.

block-check character (BCC). The BSC transmission control character that is used to determine if all of the bits that were sent were also received.

bpi. Bits per inch.

bps. Bits per second.

bracket. One or more chains of request units and their responses, representing a complete transaction, exchanged between two logical unit half-sessions. See also *RU chain*.

BSC. See *binary synchronous communications (BSC)*.

byte. A group of eight adjacent bits. In the EBCDIC coding system, one byte can represent a character. In the double-byte coding system, two bytes represent a character.

C & SM. See *communications and systems management (C & SM)*.

call level. The position of a program in a nest of programs called explicitly by the CALL instruction or implicitly by some event. The first program has a call level of 1. Any program called by a level 1 program has a call level of 2, and so on.

change authority. An object authority that allows a user to perform all operations on the object except those limited to the owner or controlled by object existence authority and object management authority. The user can add, change, and delete entries in an object, or read the contents of an entry in the object. Change authority combines object operational authority and all the data authorities.

character. Any letter, number, or other symbol in the data character set that is part of the organization, control, or representation of data.

character key. A keyboard key that allows the user to type into the system the character shown on the key. See also *function key*.

characters per inch (cpi). The number of characters printed horizontally within an inch across a page.

checksum protection. A function that protects data stored in the system auxiliary storage pool from being lost because of the failure of a single disk. When checksum protection is in effect and a disk failure occurs, the system automatically reconstructs the data when the system program is loaded after the device is repaired.

checksum set. Units of auxiliary storage defined in groups to provide a way for the system to recover data if a disk failure occurs when checksum protection is in effect.

CL. See *control language (CL)*.

class attributes. The values in the Change Job (CHGJOB) command that control the processing of routing steps in a job. These values include the run priority, time slice, delete, and default wait time-out parameters.

command. A statement used to request a function of the system. A command consists of the command name, which identifies the requested function, and parameters.

command attention (CA) key. A keyboard key that can be specified with the CA keyword to request the function specified by the keyword. Data is not returned to the system. Contrast with *command function (CF) key*.

command definition. An object that contains the definition of a command (including the command name, parameter definitions, and validity checking information) and identifies the program that performs the function requested by the command. The system-recognized identifier for the object type is *CMD.

command function (CF) key. A keyboard key that can be specified with the CF keyword to request the function specified by the keyword. Data is returned to the system. Contrast with *command attention (CA) key*.

command line. The blank line on a display where commands, option numbers, or selections can be entered.

command processing program (CPP). A program that processes a command. This program performs some validity checking and processes the command so that the requested function is performed.

communications and systems management (C & SM). A part of the system that contains the remote management support (also referred to as DHCF), the change management support (referred to as DSNX), and the problem management support (referred to as alerts).

communications controller. The I/O processor card in the card enclosure.

communications line. The physical link (such as a wire or a telephone circuit) that connects one or more work stations to a communications control unit, or connects one control unit to another. Contrast with *data link*.

completion message. A message that tells the operator when work is successfully ended.

concept. An abstract idea.

configuration. The arrangement of devices and programs that make up a data processing system. See also *system configuration.*

configure. (1) To describe the interconnected arrangement of the devices, programs, communications, and optional features installed on a system. (2) To describe setting up auxiliary storage pools and check sum protection.

consecutive processing. A method of processing in which the records in the file are read, written to, or deleted in the order in which they exist in a file. See also *random processing* and *sequential processing*.

console. A display station from which an operator can control and observe the system operation.

control block. A storage area used by a program to hold control information.

control language (CL). The set of all commands with which a user requests system functions.

control language program. A program that is created from source statements consisting entirely of control language commands.

control language variable. A program variable that is declared in a control language program and is available only to the CL program.

control panel. A panel located on the processing unit on the front of the rack that contains lights and switches to operate or service the system.

control storage. Storage in the computer that contains the programs used to control input and output operations and the use of main storage. Contrast with *main storage.*

controller. A device that coordinates and controls the operation of one or more input/output devices (such as work stations) and synchronizes the operation of such devices with the operation of the system as a whole.

controller description. An object that contains a description of the characteristics of a controller that is either directly attached to the system or attached to a communications line. The system-recognized identifier for the object type is *CTLD.

conversation. In interactive communications, the communication between the application program and a specific item (usually another application program) at the remote system.

cpi. See *characters per inch (cpi).*

creation date. The system date when an object is created. See also *job date*, and *system date.*

CTLD. See *controller description.*

current interrupted job. When a job is interrupted by pressing the Attn key, another job can be started from a command display. This job can also be interrupted by pressing the Attn key again. The current interrupted job is the most recently interrupted. The job name for the current interrupted job is displayed at the top of the Inquiry Options menu.

current library. The library that is specified to be the first user library searched for objects requested by a user. The name for the current library can be specified on the Sign-On display or in a user profile. When you specify an object name (such as the name of a file or program) on a command, but do not specify a library name, the system searches the libraries in the system part of the library list, then searches the current library before searching the user part of the library list. The current library is also the library that the system uses when you create a new object, if you do not specify a library name.

data area. A storage area used to communicate data such as CL variable values between the programs within a job and between jobs. The system-recognized identifier for the data area is *DTAARA.

data authority. A specific authority to read, add, update, or delete data. See also *add authority*, *delete authority*, *read authority*, and *update authority.*

data base. The collection of all data files stored in the system.

data base file. An object that contains descriptions of how input data is to be presented to a program from internal storage and how output data is to be presented to internal storage from a program. See also *physical file* and *logical file.*

data description specifications (DDS). A description of the user's data base or device files that is entered into the system in a fixed form. The description is then used to create files.

data file. (1) A collection of related data records organized in a specific order. (2) A file created by the specification of FILETYPE(*DATA) on the create commands.

data link. The physical connection (communications lines, modems, controllers, work stations, and other communications equipment), and the rules (protocols) for sending and receiving data between two or more locations in a data network. Contrast with *communications line.*

DBCS. See *double-byte character set (DBCS).*

DDM. See *distributed data management (DDM).*

DDM file. A file description, created by a user on the local (source) system, for a file that is stored on a remote (target) system. The DDM file provides the information needed for a local system to locate a remote system and to access the data in the remote file.

DDS. See *data description specifications.*

deallocate. To release a resource that is assigned to a specific task. Contrast with *allocate.*

dedicated service tools (DST). The part of the service function used to service the system when the operating system is not working.

dedicated system. A system intentionally reserved for a single job or task.

default. A value automatically supplied or assumed by the system or program.

default focal point. A network node that receives alerts from nodes that do not have defined focal points. Contrast with *primary focal point.*

default routing entry. The routing table entry specifying the route to be used when the table contains no explicit routing entry.

default user name. A system-provided name for a user identification for a computer system that does not want to require separate user identifications.

default value. A value supplied by the system that is used when no value is specified by the user, or the value specified by the user with the DFT keyword in DDS. See also *assumed value.*

definite response. A value in the response-requested field of the request header (RH). The value directs the receiver of the request to return a response unconditionally, whether positive or negative, to that request.

delete authority. A data authority that allows the user to remove entries from an object; for example, delete messages from a message queue or delete records from a file. Contrast with *add authority.* See also *read authority* and *update authority.*

delimiter. A character or sequence of characters that marks the beginning or end of a unit of data.

DEVD. See *device description.*

device description. Information describing a particular device that is attached to the system. The system-recognized identifier for the object type is *DEVD.

device file. A file that contains a description of how data is to be presented to a program from a device or how data is to be presented to the device from the program. Devices can be display stations, printers, a diskette unit, tape units, or a communications line.

diagnostic. Pertaining to the detection and isolation of an error.

diagnostic message. A message that contains information about errors or possible errors. This message is generally followed by an escape message.

disk. (1) A storage media made of one or more flat, circular sheets with magnetic surfaces on which information can be stored. (2) A direct-access storage medium with magnetically recorded data.

diskette writer. A function of the operating system that writes the spooled output from a program to a diskette device. See also *print writer* and *spooling writer.*

display file. A device file created by the user to support a display station.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or the information received from the system.

distributed data management (DDM). A function of the operating system that allows an application program or user on one system to use data files stored on remote systems. The systems must be connected by a communications network, and the remote systems must also be using DDM.

do group. A set of commands in a control language program defined by a DO command and an ENDDO command that is conditionally processed as a group. Contrast with *block.*

document authority. Permission granted to one user to work with documents that are owned by another user.

double-byte character set (DBCS). A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, typing, displaying, and printing DBCS characters requires hardware and supporting programs that are DBCS-capable.

DST. See *dedicated service tools (DST).*

dump. (1) (PAR) To write, at a particular instant, all or part of the contents of main or auxiliary storage onto another data medium for the purpose of protecting the data or collecting error information. (2) To copy data from main or auxiliary storage onto an external medium, such as tape, diskette, or printer.

end node. A node in an APPN network that can be a source or target node, but does not provide any routing or session services to any other node.

end-of-file delay. An interval during which the system holds a file open after the normal end of the file is reached until one or more records are updated or added to the end of the file. The length of the interval can be specified on the EOFDLY parameter.

end-of-transmission (EOT) character. The BSC transmission control character used to end transmission with the remote system.

ENQ. See *enquiry (ENQ) character.*

enquiry (ENQ) character. The BSC transmission control character usually used to request a response from the remote system or device.

EOT. See *end-of-transmission (EOT) character.*

even positive acknowledgment. See *ACK0.*

exclude authority. An object authority that prevents the user from using the object or its contents. Contrast with *all authority.*

external message queue. The part of the job message queue that sends messages between an interactive job and the work station user. For batch jobs, messages sent to the external message queue appear only in the job log.

external object. An object that has a defined object type (such as *FILE or *PGM). In general, external objects can be displayed by a user. See also *object.* Contrast with *internal object.*

field. A group of related characters (such as name or amount) that are treated as a unit on a record.

file. A generic term for the object type that refers to a data base file, a device file, or a set of related records treated as a unit. The system-recognized identifier for the object type is *FILE.

file description. The data description specifications that describe the file and its contents.

file name. The name used by a program to identify a file. See also *label.*

file separator. The pages produced at the beginning of each output file and used to separate the file from the other files being sent to an output device.

function key. A keyboard key that allows the user to select keyboard functions or programmer functions. Contrast with *character key.*

general-purpose library. The library shipped with the system that contains IBM-provided objects required for many system functions and user-created objects that are not explicitly placed in a different library when they are created. Named QGPL.

generic. Relating to, or characteristic of, a whole group or class.

generic name. The characters common to object names that can be used to identify a group of objects. A generic name ends with an * (asterisk). For example, ORD* identifies all objects whose names begin with the characters ORD.

group authority. Authority to use objects, resources, or functions from a group profile.

group data area. A data area that is automatically created when an interactive job becomes a group job. This data area is shared by all jobs in the group but cannot be used by jobs outside the group.

group job. One of up to sixteen interactive jobs that are associated in a group with the same work station device and user.

group job name. A name that is assigned to an interactive job when it is changed into a group job using the Change Group Attributes (CHGGRPA) command or when a group job is started using the Transfer to Group Job (TFRGRPJOB) command. This name is used within the group to identify the job.

group job transfer. An operation performed by the Transfer to Group Job (TFRGRPJOB) command that will either start a new group job or resume an existing group job.

group message queue. A message queue associated with a group of jobs. When the message queue is set to break or notify mode in the active group job, it is set to the same mode in any job in the group that is transferred to or to any job that gains control when the active group job is canceled.

group profile. A user profile that provides the same authority to a group of users.

hex. See *hexadecimal.*

hexadecimal. Pertaining to a numbering system with a base of 16.

host system. The primary or controlling computer in a communications network. See also *control station.*

I/O. See *input/output.*

ideographic. Pertaining to 2-byte characters consisting of pictograms, symbolic characters, and other types of symbols.

IGC. Abbreviation used in commands and keywords to represent double-byte character set functions.

independent work station. A work station that is programmable and operates independently of a host system, but can communicate with it and use selected system services. A Displaywriter is an example of an independent work station.

initial program. A user-profile program that runs when the user signs on and after the command processor program QCMD is started. QCMD calls the first program.

initial program load (IPL). The process that loads the system programs from the system auxiliary storage, checks the system hardware, and prepares the system for user operations.

initialize. To set the addresses, switches, or the contents of storage to zero, or to the starting value set by the manufacturer.

inline data file. A file described by a Data command that is included as part of a job when the job is read from an input device. The file is erased when the job ends.

input/output. Data provided to the computer or data resulting from computer processing.

input/output processor (IOP). One or more circuits of an input/output controller that processes programmed instructions, and controls one or more input/output devices or adapters.

inquiry mode. An operation during which the job currently running from a display station is interrupted so that other work can be done. The operator presses the Attn key to put the display station in inquiry mode.

interactive. Pertaining to the exchange of information between people and a computer. Contrast with *batch*.

interactive job. A job started for a person who signs on to a work station. Contrast with *batch job*.

interactive subsystem. A subsystem in which interactive jobs are processed.

interface. A shared boundary. An interface might be the hardware to connect two devices or it might be a part of main storage, or registers used by two or more computer programs.

internal object. An object that the system program uses to store the information needed to perform some system functions. Internal objects cannot be displayed by a user. For example, you cannot use a display command (like the Display Library (DSPLIB) command) to display internal objects. Contrast with *external object*.

internal storage. All main and auxiliary storage in the system.

IOP. See *input/output processor*.

IPL. See *initial program load (IPL)*.

Japanese basic-Kanji character set. A subset of Japanese DBCS, consisting of commonly used Kanji characters. There are 3,226 Kanji characters in this set.

Japanese double-byte character set. An IBM-defined double-byte character set for Japanese, consisting of the Japanese non-Kanji set, basic Kanji set, extended Kanji set, and up to 4,370 user-definable characters.

Japanese extended-Kanji character set. A subset of Japanese DBCS, consisting of less commonly used Kanji characters. There are 3,487 characters in this set.

Japanese non-Kanji character set. A subset of Japanese DBCS, consisting of non-Kanji characters like Greek, Russian, Roman numeric, alphanumeric and related symbols, Katakana, Hiragana, and special symbols. There are 550 characters in this set.

job accounting. A system function that collects information about a job's use of system resources and records that information in a journal.

job action. The network attribute that controls the handling of a job submitted from remote locations through either the SNADS network or RSCS.

job control authority. A special authority that allows a user to: change, delete, display, hold, and release all files on output queues; hold, release, and clear job queues and output queues; start writers to output queues; hold, release, change, and end other users' jobs; change the class attributes of a job; end subsystems; and start (IPL) the system. See also *all object authority*, *save system authority*, *security administrator authority*, *service authority*, and *spool control authority*.

job date. The date associated with a job. The job date usually assumes the system date, but it can be changed by the user. See also *creation date* and *system date*.

job description. A system object that defines how a job is to be processed. The object name is *JOBD.

job log. A record of requests submitted to the system by a job, the messages related to the requests, and the actions performed by the system on the job. The job log is maintained by the system program.

job message queue. A message queue that is created for each job. A job message queue receives requests to be processed (such as commands) and sends messages that result from processing the requests. A job message queue consists of an external message queue and a set of program message queues. See also *external message queue* and *program message queue*.

job name. The name of the job as identified to the system. For an interactive job, the job is assigned the name of the work station at which the job was started; for a batch job, the name is specified in the command used to submit the job. Contrast with *qualified job name*.

job queue. A list of batch jobs waiting to be started or processed by the system. The system-recognized identifier for the object type is *JOBQ.

journal. A system object used to record entries in a journal receiver when a change is made to the data base files associated with the journal. The object type is *JRN. See also *journal receiver*.

journal code. A 1-character code in a journal entry that identifies the primary category of the journal entry; for example, F identifies a file-level entry. See also *journal entry*.

journal entry. A record in a journal receiver that contains information about data base files being journaled. See also *journal code*, *journal entry identifier*, *journal entry qualifier*, and *journal entry type*.

journal entry identifier. The fields in a journal entry that identify the category of the journal entry, the type of journal entry, the date and time of the entry, the job name, the user name, and the program name.

journal entry qualifier. The part of a journal entry that identifies the name of the object for which the journal entry was created.

journal entry type. A 2-character field in a journal entry that identifies the type of user-generated or system-generated journal entry; for example, PT is the entry type for a put operation. See also *journal code* and *journal entry identifier*.

journal receiver. A system object that contains journal entries recorded when changes are made to the data in data base files or the access paths associated with the data base files. The object type is *JRNRCV. See also *journal*.

Kanji. Chinese characters used in Japanese written language.

keyed sequence access path. An access path to a data base file that is arranged according to the contents of key fields contained in the individual records. See also *arrival sequence access path* and *access path*.

Keylock feature. A security feature in which a lock and key can be used to restrict the use of the display station.

keylock switch. A switch on the control panel that can be set to one of four different positions by turning it in either a clockwise or counterclockwise direction.

keyword functions. The result of processing DDS keywords in a record format specified on an operation. See also *operation*.

label. (1) The name of a file on a diskette or tape. (2) An identifier of a command or program statement generally used for branching.

library. (1) An object on disk that serves as a directory to other objects. A library groups related objects, and allows the user to find objects by name. (2) The set of publications for a system.

library list. A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL.

licensed program. An IBM-written program that performs functions related to processing user data.

local. Pertaining to a device, system, or file that is connected directly or read directly from your system, without the use of a communications line. Contrast with *remote*.

local area network (LAN). The physical connection that allows transfer of information among devices located on the same premises.

local controller. A functional unit within the system that controls the operation of one or more directly attached input/output devices or communications lines. Contrast with *remote controller*.

local data area. A 1024-byte data area that can be used to pass information between programs in a job. A separate local data area is automatically created for each job.

local system. For interactive jobs, the system to which the display device is directly attached. For batch jobs, the system on which the job is being processed.

local work station. A work station that is connected directly to system without need for data transmission facilities. Contrast with *remote work station*.

lock state. A condition defined for an object that determines how it is locked, how it is used (read or write), and whether the object can be shared (used by more than one job).

machine interface (MI). The instruction set that tells the computer how to operate.

machine storage pool. A storage pool used by the machine and certain highly shared programs, whose size is specified in the system value QMCHPOOL.

main storage. The part of the processing unit where programs are run. Contrast with *control storage*.

main storage dump space. A section of storage reserved on the disk device that is used as a place to save main storage for recovery and debugging.

main storage pool. A division of main storage, which allows the user to reserve main storage for processing a job or group of jobs, or to use the pools defined by the system. Contrast with *auxiliary storage pool.*

megabyte. A unit of measure for storage capacity; 1 megabyte = 1 048 576 bytes.

message description. Information describing a particular message.

message identifier. A seven-character code that identifies a predefined message, and is used to get the message description from a message file. See *predefined message*.

message queue. A list on which messages are placed when they are sent to a person or program. The system-recognized identifier for the object type is *MSGQ.

MI. See *machine interface (MI)*.

microcode. An instruction or group of instructions located in storage or device controllers that controls the operation of a device or controller. Microcode cannot be called by the control program or an application program.

mode. The session limits and common characteristics of the sessions associated with advanced-program-to-program communications (APPC) devices managed as a unit with a remote location.

mode description. A system object created for advanced-program-to-program (APPC) devices that describes the session limits and the characteristics of the session, such as the maximum number of sessions allowed, maximum number of conversations allowed, the pacing value for incoming and outgoing request/response units, and other controlling information for the session.

monitor. (1) A functional unit that observes and records selected activities for analysis within a data processing system. (2) Devices or programs that observe, supervise, control, or verify system operations.

NAK. See *negative acknowledgment (NAK)* character.

negative acknowledgment (NAK) character. The BSC transmission control character that indicates that the device is not-ready or that an error occurred.

negative response. In data communications, a reply indicating that data was not received correctly or that a command was incorrect or unacceptable. Contrast with *positive response*.

network. A collection of data processing products connected by communications lines for exchanging information between stations.

network node. A node that can define the paths or routes, control route selection, and handle directory services for APPN.

node. (1) One of the systems or devices in a network. (2) A location in a communications network that provides host processing services. (3) (X.25) A point where packets are received, stored, and forwarded to another location (or data terminal equipment) according to a routing method defined for the network. (4) (APPN) See *network node* and *end node*.

object. A named unit that consists of a set of characteristics that describe the object and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed, such as programs, files, libraries, and folders.

object authority. A specific authority that controls what a system user can do to an entire object. For example, object authority includes deleting, moving, or renaming an object. There are seven types of object authorities: object operational, object management, object existence, all, change, use, and exclude.

object description. The characteristics (such as name, type, and owner name) that describe an object.

object existence authority. An object authority that allows the user to delete the object, free storage of the object, save and restore the object, transfer ownership of the object, and create an object that was named by an authority holder.

object management authority. An object authority that allows the user to specify the authority for the object, move or rename the object, and add members to data base files.

object name. The name of an object. Contrast with *qualified name*.

object operational authority. An object authority that allows the user to look at the description of an object and use the object as determined by the user's data authorities to the object. See also *all authority* and *use authority*.

object owner. A user who creates an object or to whom the ownership of an object was reassigned. The object owner has complete control over the object.

odd positive acknowledgment. See ACK1.

operating system. A collection of system programs that control the overall operation of a computer system.

operation. The result of processing statements in a high-level language. See also *keyword functions.*

output. Information or data received from a computer that is shown on a display, printed on the printer, or stored on disk, diskette, or tape.

output queue. An object that contains a list of spooled files to be written to an output device, such as a printer. The system-recognized identifier for the object type is *OUTQ.

owner. The user who creates an object (or is named the owner of an object).

packed decimal format. Representation of a decimal value in which each byte within a field represents two numeric digits except the far right byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111. Contrast with *zoned decimal format.*

packet. A data transmission information unit. A group of data and control characters, transferred as a unit, determined by the process of transmission. Commonly used data field lengths in packets are 128 or 256 bytes.

PAG. See *process access group (PAG).*

page fault. An error that occurs when a page that is marked as not in main storage is referred to by a page that is in main storage.

paging. To move a page of data between main and auxiliary storage.

parameter. A value supplied to a command or program that either is used as input or controls the actions of the command or program.

parameter list. A list of values that provide a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

physical unit. One of three types of network addressable units. A physical unit exists in each node of an SNA network to manage and monitor the resources (such as attached links and adjacent link stations) of a node, as requested by an system services control point logical unit (SSCP-LU) session.

pool. A division of main or auxiliary storage. See also *base pool, storage pool.*

positive response. A response indicating that a request arrived and was successfully received and processed. Contrast with *negative response.* See also *definite response.*

power down. An AS/400 command to turn the power off and bring an orderly end to system operation.

predefined message. A message whose description is created and stored in a message file before it is sent by the program.

primary focal point. A network node that receives alerts from nodes that the user has defined in a sphere of control. Contrast with *default focal point.*

print queue. A list of output waiting to be printed by the system.

print writer. A function of the operating system that writes the spooled output from a program to a printer. See also *diskette writer* and *spooling writer.*

printer file. A device file created by the user to support a printer device.

private authority. The authority specifically given to a user for an object that overrides any other authorities, such as the authority of a user's group profile or an authorization list. Contrast with *public authority.*

process access group (PAG). A group of objects that is primarily paged in and out of storage in a single operation when a job (process) enters a long wait.

processing. The action of performing operations and calculations on data.

processing unit. The part of the system that performs instructions and contains main storage.

profile. Data that describes the characteristics of a user, program, device, or remote location.

program message queue. An object used to hold messages that are sent between program calls of a routing step. The program message queue is part of the job message queue.

programmer subsystem. An IBM-supplied interactive subsystem used to code programs on a display station. The system object name is QPGMR.

programmer user profile. The system-supplied user profile that has the authority necessary for system and application programmers and the special authorities of save system authority and job control authority. Named QPGMR.

protocol. A set of rules controlling the communication and transfer of data between two or more devices in a communications system.

PU. See *physical unit*.

public authority. The authority given to users who do not have any specific (private) authority to an object, who are not on the authorization list (if one is specified for the object), and whose group profile has no specific authority to the object. Contrast with *private authority*.

purge. An attribute that specifies whether a job is to be marked eligible to be moved out of main storage into auxiliary storage at the end of a time slice or when entering a long wait.

QCMD. The IBM-supplied control language processor that interprets and processes CL commands for the system.

QGPL. See *general-purpose library*.

qualified job name. A job name and its associated user name and a system-assigned job number. Contrast with *job name*.

qualified name. The name of the library containing the object and the name of the object. Contrast with *object name*.

qualifier. In data processing, all names in a qualified name other than the far right, which is called the simple name.

random processing. A method of processing in which records can be read from, written to, or deleted from a file order requested by the program that is using them. See also *consecutive processing* and *sequential processing*.

read authority. A data authority that allows the user to look at the contents of an entry in an object or to run a program. See also *add authority*, *delete authority*, and *update authority*.

read/write head. The data sensing and recording unit of the diskette drive or tape drive.

reader. An internal program that reads jobs from an input device or a data base file and places them on a job queue.

recursion. An operation done in several steps, with each step using the output of the previous step.

recursion level. The position of a program in a list of programs called by the first program and any following programs. The first occurrence of a program in a job has a recursion level of 1, the second occurrence of the same program has a recursion level of 2, and so on.

recursive. Pertaining to a program or routine that calls itself after each run until it is interrupted.

recursive procedure. An active procedure that can be called from within itself or from within another active procedure.

recursive program. A program that can call itself, or be called by another program, and repeat indefinitely until a specified condition is met.

release-program-device operation. An operation that makes a program device not available for input/output operations. Contrast with *acquire-program-device operation*.

remote. Pertaining to a device, system, or file that is connected to another device, system, or file through a communications line. Contrast with *local*.

remote controller. A device or system, attached to a communications line, that controls the operation of one or more remote devices. Contrast with *local controller*.

remote device. A device whose controller is connected to an AS/400 system by a communications line.

remote work station. A work station that is connected to the system by data communications. Contrast with *local work station*.

request header. A 3-byte header preceding a request unit. Abbreviated RH. See *request/response header*. Contrast with *response header*.

request unit (RU). The record transmitted to the other system. This record can contain a request, data, or both. Contrast with *response unit (RU)*.

request/response header. Control information preceding a request/response unit that specifies the type of request/response unit and contains control information associated with that request/response unit. Abbreviated RH. See also *request unit (RU)*.

request/response unit (RU). A combined term to identify a request unit or a response unit.

resource. Any part of the system required by a job or task, including main storage, devices, the processing unit, programs, files, libraries, and folders.

response header. A header, optionally followed by a response unit, that indicates whether the response is positive or negative and that may contain a pacing response. Abbreviated RH. See *request/response header* and *request header*. See also *negative response* and *positive response*.

response unit (RU). The record sent to respond to a request. The response can be either positive or negative and can include control information. Contrast with *request unit (RU)*.

routing. The list of users who are to receive an item when it is distributed, including all users named specifically and those users named on distribution lists by the sender.

routing entry. An entry in a subsystem description that specifies the program to be called to control a routing step that runs in the subsystem.

routing step. The processing that results from running a program specified in a routing entry. Most jobs have only one routing step.

RU. See *request unit* or *response unit (RU)*.

RU chain. A set of related request/response units that are transmitted consecutively on a particular normal or expedited data flow. See also *bracket*.

save system authority. A special authority that allows the user to save and restore all objects on the system and free storage of all objects on the system. See also *all object authority*, *job control authority*, *security administrator authority*, *service authority*, and *spool control authority*.

security administrator authority. A special authority that allows a user to add users to the system distribution directory, to create and change user profiles, to add and remove access codes, and to perform office tasks, such as delete documents, folders, and document lists, and change distribution lists for other users. See also *all object authority*, *save system authority*, *job control authority*, *service authority*, and *spool control authority*.

security officer. A person assigned to control all of the security authorizations provided with the system. A security officer can, for example, remove password or resource security; or add, change, or remove security information about any system user.

sequence number. (1) The number of a record that identifies the record within the source member. (2) A field in a journal entry that contains a number assigned by the system. This number is initially 1 and is increased by 1 until the journal is changed or the sequence number is reset by the user.

sequential processing. A method of processing in which records are read, written to, or deleted in the order determined by the value of the key field. See also *consecutive processing* and *random processing*.

service authority. A special authority that allows the user to perform the alter function in the service functions. See also *all object authority*, *save system authority*, *job control authority*, *security administrator authority*, and *spool control authority*.

session. (1) The length of time that starts when a user signs on and ends when the user signs off at a display station. (2) In communications, the logical connection by which a program or device can communicate with a program or device at a remote location. (3) (SNA) A logical connection between two network locations that can be started, tailored to provide various connection protocols, and stopped, as requested. Each session is uniquely identified in a header by a pair of network addresses, identifying the origin and destination of any transmissions exchanged during the session.

SNA distribution services (SNADS). An IBM architecture that defines a set of rules to receive, route, and send electronic mail in a network of systems.

SNADS. See *SNA distribution services (SNADS)*.

source file. (1) A file of programming code that is not compiled into machine language. Contrast with *data file*. (2) A file created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications.

source system. The system that issues a request to establish communications with another system. (DDM) The system on which an application program issues a request to use a remote file. Contrast with *target system*.

special authority. The types of authority a user can have to perform system functions, including all object authority, save system authority, job control authority, security administrator authority, spool control authority, and service authority. Contrast with *specific authority*.

specific authority. The types of authority a user can be given to use the system resources, including object authorities and data authorities. See also *object authority* and *data authority*. Contrast with *special authority*.

spool. The system function of putting jobs into a storage area to wait to be printed or processed.

spool control authority. A special authority that allows the user to perform spooling functions, such as display, delete, hold, and release spooled files on the output queue for himself and other users. This authority also allows the user to change the spooled file attributes, such as the printer used to print the file. See also *all object authority*, *save system authority*, *job control authority*, *security administrator authority*, and *service authority*.

spooled file. A file that holds output data waiting to be printed, or input data waiting to be processed by the program.

spooled input file. See *inline data file*.

spooled output file. A file that causes output data to be held for later printing.

spooling. The system function that saves data in a disk file for later processing or printing.

spooling subsystem. A part of the system that provides the operating environment for the programs that read jobs onto job queues to wait for processing and write files from an output queue to an output device. IBM supplies one spooling subsystem: QSPL.

spooling writer. The generic name to refer to the function of the diskette writer and print writer.

SSCP. See *system services control point (SSCP).*

SSCP ID. A number uniquely identifying a system services control point. The SSCP ID is used in requests sent to physical units and to other system services control points.

storage pool. A logical division of storage reserved for processing a job or group of jobs.

store. To put or keep data in a storage device.

subsystem. An operating environment, defined by a subsystem description, where the system coordinates processing and resources.

synchronous data link control (SDLC). A form of communications line control that uses commands to control the transfer of data over a communications line.

synchronous processing. A series of operations that are done as part of the job in which they were requested; for example, calling a program in an interactive job at a work station. Contrast with *asynchronous processing.*

system ASP. The auxiliary storage pool where system programs and data reside. It is the storage pool used if a storage pool is not defined by the user. See also *auxiliary storage pool* and *user ASP.*

system configuration. A process that specifies the machines, devices, and programs that form a particular data processing system.

system date. The date assigned in the system values when the system is started. See also *creation date*, and *job date.*

system library. The library shipped with the system that contains objects, such as authorization lists and device descriptions created by a user; and the licensed programs, system commands, and any other system objects shipped with the system. The system identifier is QSYS.

system object. One of two machine object classifications. Any of the machine objects shipped with the system or any of the operating system objects created by the system.

system pointer. A pointer that contains addressability to a machine interface system object.

system services control point (SSCP). A focal point within an SNA network for managing the other systems and devices, coordinating network operator requests and problem analysis requests, and providing directory routing and other session services for network users.

target. (1) In advanced program-to-program communications, the program or system to which a request for processing is sent. (2) (DDM) The remote system where the request for a file is sent.

target system. In a distributed data management (DDM) network, the system that receives a request from an application program on another system to use one or more files located on the target system. Contrast with *source system.*

time slice. The amount of processor time (specified in milliseconds) allowed for a routing step before other waiting routing steps are allowed to process data.

token-ring network. A local area network that sends data in one direction throughout a specified number of locations by using the symbol of authority for control of the transmission line, called a token, to allow any sending station in the network (ring) to send data when the token arrives at that location.

transaction. In communications, an exchange between a program on a local system and a program on a remote system that accomplishes a particular action or result. See also *conversation* and *session.*

TRLAN. Abbreviation in the commands, parameters, and options for IBM Token-Ring Network. See also *token-ring network.*

uninterruptible power supply. A source of power from a battery installed between the commercial power and the system that keeps the system running, if a commercial power failure occurs, until it can complete an orderly end to system processing.

update authority. A data authority that allows the user to change the data in an object, such as a journal, a message queue, or a data area. See also *add authority*, *delete authority*, and *read authority.*

use authority. An object authority that allows the user to run a program or display the contents of a file. Use authority combines object operational authority and read authority.

user ASP. One or more auxiliary storage pools used to isolate journals, journal receivers, and save files from the other system objects stored in the system ASP. See also *auxiliary storage pool* and *system ASP.*

user ID. See *user identification (user ID).*

user ID/address. The two-part network name used in the system distribution directory and in the office applications to uniquely identify a user and send electronic mail.

user identification (user ID). (1) The name used to associate the user profile with a user when a user signs on the system. See also *user profile name*. (2) The first part of a two-part network name used in the system distribution directory and in the office applications to uniquely identify a user. The network name is usually the same as the user profile name, but does not need to be.

user profile. An object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns. It is used by the system to verify the user's authorization to read or use objects, such as files or devices, to run the jobs on the system.

user profile name. The name or code that the system associates with a user when he or she signs on the system. Also known as user ID.

work entry. An entry in a subsystem description that specifies the source from which jobs can be accepted for processing in the subsystem.

work station. A device used to transmit information to or receive information from a computer; for example, a display station or printer.

work station address. (1) A number used in a configuration file to identify a work station attached to a computer port. (2) The address to which the switches on a work station are set, or the internal address assumed by the system, if no address is specified.

work station controller. An I/O controller card in the card enclosure that provides the direct connection of local work stations to the system.

work station entry. An entry in a subsystem description that specifies the work stations from which users can sign on to the subsystem or from which interactive jobs can transfer to the subsystem.

work station user profile. The system-supplied user profile that has the authority required by work station operators. Named QUSER.

writer. (1) The part of the operating system spooling support that writes spooled output to an output device independently of the program that produced the output. (2) (RJE) A program that receives output data (files) from the host system.

X.25. In data communications, a specification of the CCITT that defines the interface to an X.25 (packet-switching) network.

zoned decimal format. A format for representing numbers in which the digit is contained in bits 4 through 7 and the sign is contained in bits 0 through 3 of the far right byte; bits 0 through 3 of all other bytes contain 1s (hex F). For example, in zoned decimal format, the decimal value of +123 is represented as 1111 0001 1111 0010 1111 0011. Same as unpacked decimal format. Contrast with *packed decimal format*.

Index

Special Characters

***BASE**
- activity level 5-18
- pool size 5-18

A

abnormal system ending
- preventing lost journal entries 8-18

accounting code assignment areas
- job description 8-20
- user profile 8-20

accounting code information, IPL 8-18

accounting segment, illustration 8-8

active job
- description 2-45

activity level 5-7
- *BASE 5-18
- adjusting 5-19
- description 2-45
- job queue entry 2-45
- QINTER 5-18
- QSPL 5-17
- routing entry 2-45
- storage pool 2-45
- subsystem 2-22, 2-45
- summary of 2-45
- system 2-45
- work station entry 2-45

Add Autostart Job Entry (ADDAJE) command 2-3

Add Job Queue Entry (ADDJOBQE) command
- use 2-3

Add Routing Entry (ADDRTGE) command
- interactive example 2-48
- use 2-3

Add Work Station Entry (ADDWSE) command
- example 2-48
- use 2-3

ADDAJE (Add Autostart Job Entry) command 2-3

adding
- job queue entry 2-49
- routing entry 2-49
- work station entry 2-48

ADDJOBQE (Add Job Queue Entry) command
- use 2-3

ADDRTGE (Add Routing Entry) command
- interactive example 2-49
- use 2-3

ADDWSE (Add Work Station Entry) command
- example 2-48
- use 2-3

alert controller description 7-28

alert network attributes
- ALRCTLD 7-28
- ALRDFTFP 7-28
- ALRLOGSTS 7-28
- ALRPRIFP 7-28
- ALRSTS 7-28

alert status 7-28

allocation
- shipped objects for 1-21
- system values
  - QACTJOB 7-22
  - QADLACTJ 7-23
  - QADLSPLA 7-23
  - QADLTOTJ 7-23
  - QJOBMSGQSZ 7-24
  - QJOBMSGQTL 7-24
  - QJOBSPLA 7-24
  - QTOTJOB 7-24

ALRCTLD network attribute 7-28

ALRDFTFP network attribute 7-28

ALRLOGSTS network attribute 7-28

ALRPRIFP network attribute 7-28

ALRSTS network attribute 7-28

APPN network attributes
- DFTMODE network attribute 7-28
- LCLCPNAME network attribute 7-28
- LCLLOCNAME network attribute 7-28
- LCLNETID network attribute 7-28
- MAXINTSSN network attribute 7-28
- MAXLOCCNV network attribute 7-28
- NETSERVER network attribute 7-28
- NODETYPE network attribute 7-28
- RAR network attribute 7-28

AT parameter
- controlling subsystem 2-47
- prevent Sign On prompt 4-8

Attention key
- call levels and 3-9
- conditions for using 3-11
- programming for 3-1

Attention key handling programs
- description 3-1
- guidelines 3-12

autostart job
- description 1-2, 2-52
- using to start a job 2-36

autostart job entry
- defining 2-7
- description 1-4

auxiliary storage
- system values 7-26

B

base storage pool (*BASE)
- See also performance tuning
- description 2-4

batch job
- See also job
- considerations 2-35
- description 1-1
- multiple pools for 5-22
- rerouting 2-43
- sending completion messages for 4-5
- starting 2-31
- submitting from another job 2-37
- transferring 2-43
- use of shipped work management objects 1-18
- ways started 2-31

Batch Job (BCHJOB) command 1-19

batch job starting and routing illustrations
- description 2-31
- QCL controls routing step for job submitted using SBMJOB command 2-34
- QCMD controls routing step for job submitted through spooling reader 2-32

batch subsystem
- for nighttime jobs, example 4-6
- how job queue is identified 1-18
- placing jobs on a job queue 1-19

BCHJOB (Batch Job) command
- example 2-14
- use 1-19

C

CALL (Call) command
- as RQSDTA on SBMJOB command 2-37

Change Autostart Job Entry (CHGAJE) command 2-3

Change Job (CHGJOB) command
- example 4-4
- use 1-4

Change Job Queue Entry (CHGJOBQE) command 2-3

Change Library List (CHGLIBL) command
- example 4-4

Change Network Attributes (CHGNETA) command 7-28, 7-29

change priority and time slice 4-9

Change Routing Entry (CHGRTGE) command 2-3

Change Subsystem Description (CHGSBSD) command 2-3

Change System Value (CHGSYSVAL) command
- example 7-7
- use 7-7

Change Work Station Entry (CHGWSE) command 2-3

changing
- network attributes 7-29
- subsystem descriptions 2-3
- system value 7-7

changing class attributes 2-16

changing priority and time slice 4-9

Check Record Locks (CHKRCDLCK) command 3-3

CHGACGCDE command, allowing use of 8-16

CHGAJE (Change Autostart Job Entry) command 2-3

CHGJOB (Change Job) command
- example 4-4
- use 1-4

CHGJOBQE (Change Job Queue Entry) command 2-3

CHGLIBL (Change Library List) command
- example 4-4

CHGNETA (Change Network Attributes) command 7-28, 7-29

CHGRTGE (Change Routing Entry) command 2-3

CHGSBSD (Change Subsystem Description) command 2-4

CHGSYSVAL (Change System Value) command
- example 7-7
- use 7-7

CHGWSE (Change Work Station Entry) command 2-3

CHKRCDLCK (Check Record Locks) command 3-3

class
- See also object
- contents 2-15
- creating 2-15
- defining 2-15
- deleting 2-16
- description 2-15, 2-22
- displaying 2-15
- relationship to subsystem description 2-22

command 1-19
- Batch Job (BCHJOB) command 1-19
- BCHJOB (Batch Job) command 1-19
- Change Job (CHGJOB) 4-4
- Change Job (CHGJOB) command 1-4
- Check Record Locks (CHKRCDLCK) command 3-3
- CHGJOB (Change Job) 4-4
- CHGJOB (Change Job) command 1-4
- CHKRCDLCK (Check Record Locks) command 3-3
- DMPTRC (Dump Trace) 6-3
- Dump Trace (DMPTRC) 6-3
- End Group Jobs (ENDGRPJOB) command 3-3
- End Performance Monitor (ENDPFRMON) 6-3
- End Subsystem (ENDSBS) command 1-6
- ENDGRPJOB (End Group Job) command 3-3
- ENDPFRMON (End Performance Monitor) 6-3
- ENDSBS (End Subsystem) command 1-6
- processor (QCMD) 1-7
- Reroute Job (RRTJOB) command 1-5
- Retrieve Group Attributes (RTVGRPA) 3-6
- RRTJOB (Reroute Job) command 1-5
- RTVGRPA (Retrieve Group Attributes) 3-6
- SBMDBJOB (Submit Data Base Jobs) command 1-19
- SBMDKTJOB (Submit Diskette Job) command 1-19
- SBMJOB (Submit job) command 1-19

command *(continued)*
  Set Attention Program (SETATNPGM) command 3-9
  SETATNPGM (Set Attention Program) command 3-9
  Start Performance Monitor (STRPFRMON) 6-2
  STRPFRMON (Start Performance Monitor) 6-2
  Submit Data Base Jobs (SBMDBJOB) command 1-19
  Submit Diskette Jobs (SBMDKTJOB) command 1-19
  Submit Job (SBMJOB) command 1-19
  TFRGRPJOB (Transfer to Group Job) command 3-3
  TFRJOB (Transfer Job) command 1-5
  Transfer Job (TFRJOB) command 1-5
  Transfer to Group Job (TFRGRPJOB) command 3-3
  Work with Active Jobs (WRKACTJOB) 5-13
  Work with System Status (WRKSYSSTS) 5-10
  WRKACTJOB (Work with Active Jobs) 5-13
  WRKSYSSTS (Work with System Status) 5-10
commands
  list of display 2-50
communications data
  collecting 6-2
communications device allocation 1-21
  rules for 1-21
communications entry
  activity level 2-45
  description 1-4, 2-52
communications job
  job description 2-52
  routing data for 1-5
  security considerations 2-52
communications space table 5-16
controlling a group of work stations 4-9
controlling subsystem
  See also QBASE
  as shipped by IBM 1-7
  creating another 2-47
  description 1-7
Create Class (CRTCLS) command
  example 2-48
  use 2-15
Create Job Description (CRTJOBD) command
  example 2-48
  use 2-14
Create Job Queue (CRTJOBQ) command
  example 4-6
  use 2-19
Create Output Queue (CRTOUTQ) command
  example 4-4
  use 2-19
Create Subsystem Description (CRTSBSD) command
  for controlling subsystem 2-47
  interactive example 2-48
  use 2-3
creating
  class 2-15
  controlling subsystem 2-47
  interactive subsystem description 2-48
  job description 2-14
  job queue 2-19
  programmer output queue 4-4
  subsystem description
    controlling 2-47
    interactive 2-48
CRTCLS (Create Class) command
  example 2-48
  use 2-15
CRTJOBD (Create Job Description) command
  example 2-48
  use 2-14
CRTJOBQ (Create Job Queue) command
  example 4-6
  use 2-19
CRTOUTQ (Create Output Queue) command
  example 4-4
  use 2-19
CRTSBSD (Create Subsystem Description) command
  for controlling subsystem 2-47
  interactive example 2-48
  use 2-3
currency symbol
  description 7-10

D

data
  communications 6-1
  system 6-1
  trace 6-1
data collection interval
  internal 6-4
data file
  QAPMASYN 6-29
  QAPMBSC 6-30
  QAPMCIOP 6-39
  QAPMCONF 6-9
  QAPMDIOP 6-41
  QAPMDISK 6-22
  QAPMECL 6-36
  QAPMHDLC 6-27
  QAPMJOBS 6-19
  QAPMLIOP 6-44
  QAPMPOOL 6-26
  QAPMRESP 6-46
  QAPMSYS 6-10
  QAPMX25 6-33
date constant
  QDATE 7-8
  QDATFMT system value 7-10
  QDATSEP system value 7-10
date format
  QDATFMT system value 7-10

date separator, system value 7-10
date system values 7-8
DDM network attributes
  DDMACC 7-29
DDMACC network attribute 7-28
default maximum wait time 2-15
Delete Subsystem Description (DLTSBSD) command 2-4
deleting
  class 2-16
  subsystem description 2-4
device allocation
  communications 1-21
    rules for 1-21
DFTMODE network attribute 7-28
DFTWAIT parameter 2-15
Display Job (DSPJOB) command 2-50
Display Job Description (DSPJOBD) command 2-50
Display Network Attributes (DSPNETA) command 2-50, 7-29
display stations
  mixing double-byte and alphameric 4-9
Display Submitted Jobs (DSPSBMJOB) command 2-50
Display Subsystem (DSPSBS) command 2-50
Display System (DSPSYS) command 2-50
Display System Value (DSPSYSVAL) command 7-6
Display System Value (DSYSYSVAL) command 2-50
displaying
  class 2-16
  system value 7-6
displays
  group job selection 3-17
  system value 7-6
  work management 2-50
DLTSBSD (Delete Subsystem Description) command 2-4
DMPTRC (Dump Trace) command 6-3
double-byte request data
  example of using 4-3
DSPJOB (Display Job) command 2-50
DSPJOBD (Display Job Description) command 2-50
DSPNETA (Display Network Attributes) command 2-50, 7-29
DSPSBMJOB (Display Submitted Jobs) command 2-50
DSPSBS (Display Subsystem) command 2-50
DSPSYS (Display System) command 2-50
DSPSYSVAL (Display System Value) command 2-50, 7-6
Dump Trace (DMPTRC) command 6-3
dynamic menu 3-13

E

editing system values
  QCURSYM 7-10
  QDATFMT 7-10
  QDATSEP 7-10
  QDECFMT 7-10
End Group Jobs (ENDGRPJOB) command 3-3
End Job (ENDJOB) command 1-19
end of job display 2-30
End Performance Monitor (ENDPFRMON) command 6-3
End Subsystem (ENDSBS) command 1-6
ENDGRPJOB (End Group Job) command 3-3
ending, prevent 4-3
ENDJOB (End Job) command 1-19
ENDPFRMON (End Performance Monitor) command 6-3
ENDSBS (End Subsystem) command 1-6
examples
  adding
    routing entry 2-49
    work station entry 2-49
  adding a second job queue 4-5
  changing work management objects 4-1
  controlling programs for user sign-ons 4-1
  creating
    class 2-48
    controlling subsystem description 2-47
    interactive subsystem description 2-48
    job description 2-48
    programmer output queue 4-4
    subsystem description for nighttime jobs 4-6
  displaying
    system value 7-6
  mixing double-byte and alphameric display stations 4-9
  perform long-running functions from work station 4-8
  request data, using 4-3
  routing data, using 4-2
  submitting a job 2-37
  using double-byte request data 4-3
  using request data 4-3
  using routing data 4-2
examples, programming
  changing user account code 8-21
  submitting a job using parameters from a display file 2-39
  time-dependent scheduling 4-11

F

fixed menu 3-13

functional space table 5-17

G

graphic character set
- system value QCHRID 7-11

group jobs
- advantages of 3-2
- attention key handling menu 3-13
  - approach for programmers 3-19
  - combination approach 3-19
  - dynamic menu 3-17
  - fixed menu 3-13
- attention key handling programs guidelines 3-12
- concepts of 3-2
- description 3-1
- performance 3-24
- relationship of group jobs to a secondary interactive job 3-4
- returning to the normal group job 3-23
- sample application 3-5
- starting, handling, and ending 3-3
- transferring to another group job without seeing a menu 3-23

guidelines for performance 5-1

I

ineligible queue 5-7

initial menu
- using 4-2

initial program
- not routing to QCMD 2-31
- QOPRMENU for QSYSOPR 1-7
- routing to QCMD 2-31
- using 4-2

initial system values 7-2

interactive job
- See also job
- considerations 2-30
- multiple pools for 5-21
- rerouting 2-43
- starting and routing illustrations
  - direct routing to user program based on user 2-27
  - direct routing to user program based on work station 2-28
  - routing based on receiving a communications program start request 2-29
  - routing to QCL based on work station 2-25
- transferring 2-43
- use of shipped work management objects 1-10
- using initial programs 2-31
- using PURGE parameter for 5-21

interactive subsystem
- See also QTCL, QINTER, QPGMR, subsystem

interactive subsystem *(continued)*
- creating description 2-48
- example 2-48

interactive subsystem, spooling subsystem′.subsystem
- See also batch subsystem, controlling subsystem, summary 1-22

IPL tuning 5-14

J

job
- See also job description, job message queue
- batch job considerations 2-35
- description of 1-3
- displaying a job 2-50
- interactive job considerations 2-30
- objects used by 5-5
- placing on a job queue
  - batch 1-19
  - spooling 1-20
- priority
  - description 2-14
- rerouting 2-43
- sources of starting 1-4
- states 5-2
- submitting from another job 2-37
- transferring 2-43

job accounting
- abnormal ends 8-17
- accounting codes 8-7
- accounting codes length 8-7
- accounting journal 8-15
- ACGCDE parameter 8-7
- analyzing data 8-16
- and QHST messages, comparison 8-6
- change the system value QACGLVL 8-14
- converting job accounting journal entries 8-19
- CPF1164 comparison 8-6
- creating a journal receiver 8-14
- DP accounting journal information 8-11
- DSPJRN command 8-19
- JB accounting journal information 8-9
- job description 8-21
- job description parameter 8-7
- job log creation 8-9
- QACGLVL value, changing 8-18
- recovery considerations 8-17
- security 8-16
- security officer 8-16
- segments 8-8
- SP accounting journal information 8-12
- steps to start 8-14

job accounting, setting up 8-14

job action (JOBACN) artwork attribute 7-28

job description 2-14
- See also object
- changing 2-15
- contents 2-14

job description *(continued)*
  creating 2-14
  damaged 2-15
  deleting 2-15
  description 1-3
  displaying 2-15
  identifying 1-4
  overriding 1-4
  relationship to subsystem description 2-22
  security considerations 2-15
  use of 2-15
job name
  qualified 1-14
job queue
  See also object
  adding a second 4-5
  creating 2-19
  defining 2-19
  identified to batch subsystem 1-18
  identified to spooling subsystem 1-20
  multiple 2-8
job queue entry
  activity level 2-45
  adding 2-49
  contents 2-7
  description 1-4
job space 5-15
job starting and routing
  autostart job illustration 2-36
  batch job considerations 2-35
  batch job illustrations 2-31
  interactive job considerations 2-30
  interactive job illustrations 2-24
  rerouting jobs 2-43
  sources of 1-4
  submitting a job from another job 2-37
  summary of activity level controls 2-45
  transferring jobs 2-43
job state 5-2
  ineligible 5-2
  wait 5-2
job transition
  active-to-ineligible 5-12
  wait-to-ineligible 5-12
JOBACN network attribute 7-28

L

LCLCPNAME network attribute 7-28
LCLLOCNAME network attribute 7-28
LCLNETID network attribute 7-28
level
  activity 5-7
library list
  changing
    example 4-4
  specifying in job description 2-14
  system values 7-22
lock conflict 5-8

M

machine pool
  communications space in 5-16
  functional space in 5-17
  job space in 5-15
machine pool size
  setting initial 5-15
machine running priority
  description 2-15
machine storage pool
  description 2-4
MAXHOP network attribute 7-28
maximums
  activity levels 2-45
  instruction wait time, default 2-15
  jobs 2-45
  number of pools 1-6
  processing unit time 2-15
  temporary storage 2-15
MAXINTSSN network attribute 7-28
MAXLOCCNV network attribute 7-28
menu
  dynamic 3-13
  fixed 3-13
  program call 1-10
message and logging size system values 7-25
message queue
  default for object distribution 7-28
MSGQ network attribute 7-28

N

NETSERVER network attribute 7-28
network attributes
  ALRCTLD 7-28
  ALRDFTFP 7-28
  ALRLOGSTS 7-28
  ALRPRIFP 7-28
  ALRSTS 7-28
  changing 7-29
  DDMACC 7-28
  description 1-3, 7-28
  DFTMODE 7-28
  displaying 7-29
  JOBACN 7-28
  LCLCPNAME 7-28
  LCLLOCNAME 7-28
  LCLNETID 7-28
  MAXHOP 7-28
  MAXINTSSN 7-28
  MAXLOCCNV 7-28
  MSGQ 7-28
  NETSERVER 7-28
  NODETYPE 7-28
  OUTQ 7-28
  PCSACC 7-28

network attributes *(continued)*
  RAR 7-28
  SYSNAME 7-28
NODETYPE network attribute 7-28

O

objects
  used by jobs 5-5
output priority
  description 2-14
output queue
  See also object
  creating for programmer 4-4
  default for object distribution 7-28
  defining 2-19
OUTQ network attribute 7-28

P

PAG (process access group) 5-5
  transfer 5-6
page fault rate 5-11
page faulting 5-11
PC Support network attributes
  PCSACC 7-29
PCSACC network attribute 7-28
perform long-running functions from work station 4-8
performance
  activity levels 5-7
  adjusting activity level 5-19
  adjusting pool size 5-19
  basic tuning 5-14
  choosing pool configuration 5-17
  commands 5-10
  data
    communications 6-1
    system 6-1
    trace 6-1
  group jobs 3-24
  guidelines 5-1
  IPL tuning 5-14
  overview 5-2
  setting machine pool size 5-15
  specialized tuning 5-20
  Submit Data Base Job (SBMDBJOB) command 2-18
  Submit Diskette Job (SBMDKTJOB) command 2-18
  time slice 5-9
  using PURGE parameter 5-6
performance data
  collecting 6-1, 6-2
    preparation for 6-1
  files, content of 6-8
  QAPMASYN file 6-29
  QAPMBSC file 6-30
  QAPMCIOP file 6-39
  QAPMCONF file 6-9
performance data *(continued)*
  QAPMDIOP file 6-41
  QAPMDISK file 6-22
  QAPMECL file 6-36
  QAPMHDLC file 6-27
  QAPMJOBS file 6-19
  QAPMLIOP file 6-44
  QAPMPOOL file 6-26
  QAPMRESP file 6-46
  QAPMSYS file 6-10
  QAPMX25 file 6-33
  system and communications, collecting 6-2
  system, collecting 6-2
  trace, collecting 6-2
pool
  multiple 5-21, 5-22
pool configuration
  choosing your 5-17
pool size
  *BASE 5-18
  adjusting 5-19
  machine 5-15
  QSPL 5-17
powering down system in unattended environment 4-6
printer file accounting 8-3
priority, running
  changing 2-16, 4-9
process access group 5-5
  transfer 5-6
processing unit
  maximum time 2-15
program start request 1-5
programmer output queue, creating 4-4
programming examples
  submitting a job using parameters from a display file 2-39
  time-dependent scheduling 4-11
purge
  description 2-15
PURGE parameter 5-6
  on interactive job 5-21

Q

QABNORMSW system value 7-3
QACGLVL system value 7-25
  initial 7-5
QACTJOB system value
  description 7-22
  initial 7-5
  lower limit 7-22
QADLACTJ system value
  description 7-23
  initial 7-5
  lower limit 7-23
QADLSPLA system value
  description 7-23
  initial 7-5

QADLSPLA system value *(continued)*
  lower limit 7-23
QADLTOTJ system value
  description 7-23
  initial 7-5
  lower limit 7-23
QAUTOCFG system value 1-8, 7-11
  initial 7-3
QBASACTLVL system value
  description 7-26
  initial 7-6
  lower limit 7-26
QBASE
  description of controlling subsystem 1-7
QBASE subsystem 1-9
QBASPOOL system value
  See also base storage pool
  description 7-26
  initial 7-6
QBATCH
  use of shipped objects for batch jobs 1-18
QBATCH subsystem 1-9
QCHRID system value
  description 7-11
  initial 7-11
QCL (command processor)
  considerations for using, for interactive jobs 2-30
QCMD (command processor)
  description 1-7
  routing to, based on work station 2-25
  using to control routing step for batch job 2-32
QCMN subsystem 1-9
QCMNRCYLMT system value 7-12
QCONSOLE system value
  initial 7-3
QCTL subsystem 1-9
QCTLSBSD system value 1-8
  description 7-12
  initial 7-3
QCURSYM system value
  description 7-10
  initial 7-2
QDATE system value 7-8
  initial 7-2
QDATFMT system value
  description 7-10
  initial 7-2
QDATSEP system value
  description 7-10
  initial 7-2
QDAY system value 7-8
  initial 7-2
QDBRCVYWT system value
  description 7-13
  initial 7-3
QDECFMT system value
  description 7-10
  initial 7-2
QDEVNAMING system value 1-8, 7-13
QHOUR system value 7-9
  initial 7-2
QHST messages, contrasted with job accounting 8-6
QHSTLOGSIZ system value
  description 7-25
  initial 7-5
  lower limit 7-25
QIGC system value 7-3, 7-13
QINTER
  activity level for 5-18
  prevent sign on prompt from being displayed 4-8
  subsystem
    See also subsystem
    used for security officer sign-on 1-13
QINTER subsystem 1-9
QIPLDATTIM system value 7-13
  initial 7-3
QIPLSTS system value 7-3, 7-14
QIPLTUNE 5-14
QIPLTYPE system value 1-8, 7-14
  initial 7-3
QJOBMSGQSZ system value
  description 7-24
  initial 7-5
  lower limit 7-24
QJOBMSGQTL system value
  description 7-24
  initial 7-5
  lower limit 7-24
QJOBSPLA system value
  description
  initial 7-5
  lower limit
QKBDTYPE system value
  description 7-14
  initial 7-3
QLEAPADJ 7-8
QLEAPADJ system value
  initial 7-2
QLUS system job 1-21
QMAXACTLVL system value
  description 7-26
  initial 7-6
  lower limit 7-26
QMAXSIGN system value 7-15
QMCHPOOL system value
  See also machine storage pool
  description 7-27
  initial 7-6
QMINUTE system value 7-9
  initial 7-2
QMONTH system value 7-8
  initial 7-2
QPFRADJ system value 7-16
QPGMR user profile 1-2
QPRTDEV system value 7-3, 7-16
  initial 7-3

QPRTTXT system value
- description 7-25
- initial 7-5

QPWRDWNLMT system value
- description 7-17
- initial 7-3
- lower limit 7-17

QPWRRSTIPL system value 7-17
- initial 7-3

QRMTIPL system value 7-17
- initial 7-3

QSCPFCONS system value
- description 7-17
- initial 7-3

QSECOFR user profile 1-2

QSECOND system value 7-9
- initial 7-2

QSECURITY system value 1-8, 7-17

QSPCENV system value 1-8, 7-18

QSPL
- activity level 5-17
- pool size 5-17
- use of shipped objects for spooling jobs 1-20

QSPL subsystem 1-9

QSRLNBR system value 7-18

QSRVDMP system value 7-26
- initial 7-5

QSTRPRTWTR system value 7-19
- initial 7-3

QSTRUPPGM system value 7-18
- initial 7-3

QSYSARB 2-9

QSYSLIBL system value
- description 7-22
- initial 7-5

QSYSOPR user profile 1-2

QTIME system value 7-9
- initial 7-2

QTOTJOB system value
- description 7-24
- initial 7-5
- lower limit 7-24

qualified job name 1-14

queue
- ineligible 5-7
- output 2-19

QUPSDLYTIM system value 7-19
- description 7-20

QUPSMSGQ system value
- description 7-21

QUSER user profile

QUSRLIBL system value
- description 7-22
- initial 7-5

QYEAR system value 7-8
- initial 7-2

R

RAR network attribute 7-28

recovery considerations
- time-dependent scheduling 4-16

Remove Autostart Job Entry (RMVAJE) command 2-3

Remove Job Queue Entry (RMVJOBQE) command 2-3

Remove Routing Entry (RMVRTGE) command 2-3

Remove Work Station Entry (RMVWSE) command 2-3

request data
- description 2-33
- example of using 4-3
- in job description 2-14

Reroute Job (RRTJOB) command
- reasons for rerouting 2-43
- use 1-5

rerouting jobs 2-43

resource accounting 8-2, 8-8

Retrieve Group Attributes (RTVGRPA) command, example 3-6

Retrieve Network Attributes (RTVNETA) command 7-29

Retrieve System Value (RTVSYSVAL) command
- use 7-1

retrieving
- system value 7-1

RMVAJE (Remove Autostart Job Entry) command 2-3

RMVJOBQE (Remove Job Queue Entry) command 2-3

RMVRTGE (Remove Routing Entry) command 2-3

RMVWSE (Remove Work Station Entry) command 2-3

routing based on communications program start request
- based on communications program start request 2-29

routing data
- See also routing entry, routing keywords, routing step
- description 1-5
- example of using 4-2
- for batch jobs 1-18
- in job description 2-14
- specifying 2-24

routing directly to user program
- based on user 2-27
- based on work station 2-28
- to control job submitted using SBMJOB command 2-34

routing entry
- See also routing data, routing step
- activity level 2-45
- adding 2-48
- contents 2-12
- description 1-3, 2-12

routing step
- See also routing data, routing entry
- activity for display of command entry display 1-15
- description 1-3
- for batch jobs 1-18
- relationship to parts of a subsystem description 1-5
- starting 1-5

routing step *(continued)*
  using QCMD to control 2-32
  where run 1-6
routing to QCMD
  based on work station 2-25
  to control job submitted through spooling reader 2-32
RQSDTA parameter
  using a variable on 2-38
RRTJOB (Reroute Job) command
  reasons for rerouting 2-43
  use 1-5
RTVGRPA (Retrieve Group Attributes) command 3-6
RTVNETA (Retrieve Network Attributes) command 7-29
RTVSYSVAL (Retrieve System Value) command
  use 7-1

S

sample commands
  SBMTIMJOB 4-12, 4-23
  SNDTIMMSG 4-11, 4-18
SBMDBJOB (Submit Data Base Jobs) command 1-19, 2-17
SBMDKTJOB (Submit Diskette Job) command 1-19, 2-17
SBMJOB (Submit job) command
  example 2-37
  use 1-19, 2-37
scheduling priority 2-14
security considerations
  time-dependent scheduling 4-16
security considerations with job description
  autostart jobs 2-52
  batch jobs 2-51
  communications job 2-52
  interactive job 2-51
security officer
  sign-on 1-12
Set Attention Program (SETATNPGM) command 3-9
SETATNPGM (Set Attention Program) command 3-9
setting up job accounting 8-14
shared storage pool 1-6
shipped system characteristics 1-7
sign-off, preventing 4-3
sign-on display file
  changing 2-11
signing on
  controlling 4-1
  under QSECOFR user profile 1-12
  under security officer user profile 1-12
  under work station user profile 1-10
  with SECOFR password 1-12
SNA network attributes
  JOBACN 7-29
  MAXHOP 7-29
  MSGQ 7-29
SNA network attributes *(continued)*
  OUTQ 7-29
special authority
  *JOBCTL 2-16
spooling jobs
  description 1-2
  queues used 2-17
  use of shipped objects 1-20
spooling subsystem
  how job queue is identified 1-20
  placing jobs on a job queue 1-20
Start Performance Monitor (STRPFRMON) command 6-2
Start Subsystem (STRSBS) command 1-6
starting a routing step 1-5
starting a subsystem 1-6
starting line numbers, variable 1-7
state
  key/think wait 5-4
  long wait 5-3
  short wait 5-3
storage pool 5-5
  See also base storage pool, machine storage pool
  activity level
    description 1-6, 2-45
  definitions 2-4
  description 1-6
  for routing step running 1-6
  maximum number 1-6
  size 1-6
storage system values 7-26
STRPFRMON (Start Performance Monitor) command 6-2
STRSBS (Start Subsystem) command 1-6
Submit Data Base Jobs (SBMDBJOB) command 1-19, 2-17
Submit Diskette Jobs (SBMDKTJOB) command 1-19, 2-17
Submit Job (SBMJOB) command
  compared to SBMxxxJOB commands 2-17
  example 2-37
  use 1-19, 2-37
  using CALL command as request data 2-38
submitting a batch job from another job 2-37
submitting a job using parameters from a display file 2-39
subsystem
  See also batch subsystem, controlling subsystem, interactive subsystem, spooling subsystem
  activity
    creation of a job for security officer 1-13
    display of command entry display 1-15
    display of program call menu 1-10
    starting a routing step 1-15
  activity level
    description 2-45
  controlling
    See also QBASE
    as shipped by IBM 1-7

subsystem *(continued)*
  - controlling *(continued)*
    - changing 2-47
    - description 1-7
  - creating
    - controlling 2-47
    - interactive example 2-48
  - defining another controlling 2-47
  - description of 1-3
  - ending a 1-6
  - examples 2-48
  - QBASE 1-9
  - QBATCH 1-9
  - QCMN 1-9
  - QCTL 1-9
  - QINTER 1-9
  - QSPL 1-9
  - starting a 1-6

subsystem configurations
  - shipped 1-9

subsystem description
  - See also object, QBASE, QBATCH, QCTL, QINTER, QSPL
  - attributes 2-3
  - changing 2-3
  - class, relationship to 2-21
  - contents 2-3
  - creating for nighttime jobs 4-6
  - defining storage pools 2-3
  - deleting 2-3
  - description 1-3
  - displaying 2-3
  - job description, relationship to 2-21
  - relationship of commands used 2-20
  - storage pool definitions 2-3

summaries
  - controls on activity levels 2-45
  - effect of shipped objects on the system 1-17
  - work management concepts and terms 1-2

SYSNAME network attribute 7-28

system
  - See also system value
  - activity level 2-45
  - control system values
    - QABNORMSW 7-3
    - QCTLSBSD 7-12
    - QDBRCVYWT 7-13
    - QIPLSTS 7-3
    - QPFRADJ 7-3
    - QPWRDWNLMT 7-17
    - QSCPFCONS 7-17
  - date
    - description 7-8
    - format 7-8
  - decimal format 7-3
  - time 7-9

system arbiter 2-9

system control program in base pool 2-4

system data
  - collecting 6-2
  - collecting only 6-2

system job
  - QLUS 1-21

system menu 1-7

system name (SYSNAME network attribute)
  - description 7-28

system value
  - accessing in CL program 7-1
  - allocation 7-22
  - changing 7-7
  - date 7-8
  - description 1-3
  - displaying 7-6
  - editing 7-10
  - initial values 7-2
  - library list 7-22
  - lists 7-1
  - logging 7-25
  - message 7-25
  - QABNORMSW 7-3, 7-11
  - QACGLVL 7-5, 7-25
  - QACTJOB 7-5, 7-22
  - QADLACTJ 7-5, 7-23
  - QADLSPLA 7-5, 7-23
  - QADLTOTJ 7-5, 7-23
  - QAUTOCFG 1-8, 7-3, 7-11
  - QBASACTLVL 7-6, 7-26
  - QBASPOOL 7-6, 7-26
  - QCHRID 7-3, 7-11
  - QCMNRCYLMT 7-3, 7-12
  - QCONSOLE 7-3, 7-12
  - QCTLSBSD 1-8, 7-3, 7-12
  - QCURSYM 7-2, 7-10
  - QDATE 7-2, 7-8
  - QDATFMT 7-2, 7-10
  - QDATSEP 7-2, 7-10
  - QDAY 7-2, 7-8
  - QDBRCVYWT 7-3, 7-13
  - QDECFMT 7-2, 7-10
  - QDEVNAMING 1-8, 7-3, 7-13
  - QHOUR 7-2, 7-9
  - QHSTLOGSIZ 7-5, 7-25
  - QIGC 7-3, 7-13
  - QIPLDATTIM 7-3, 7-13
  - QIPLSTS 7-3, 7-11, 7-14
  - QIPLTUNE 5-14
  - QIPLTYPE 1-8, 7-3, 7-14
  - QJOBMSGQSZ 7-5, 7-24
  - QJOBMSGQTL 7-5, 7-24
  - QJOBSPLA 7-5, 7-24
  - QKBDTYPE 7-3, 7-14
  - QLEAPADJ 7-2, 7-8
  - QMAXACTLVL 7-6, 7-26
  - QMAXSIGN 7-3, 7-15
  - QMCHPOOL 7-6, 7-27

system value *(continued)*
- QMINUTE 7-2, 7-9
- QMONTH 7-2, 7-8
- QPFRADJ 7-3, 7-16
- QPRTDEV 7-3, 7-16
- QPRTTXT 7-5, 7-25
- QPWRDWNLMT 7-3, 7-17
- QPWRRSTIPL 7-3, 7-17
- QRMTIPL 7-3, 7-17
- QSCPFCONS 7-3, 7-17
- QSECOND 7-2, 7-9
- QSECURITY 1-8, 7-3, 7-17
- QSPCENV 1-8, 7-3, 7-18
- QSRLNBR 7-3, 7-18
- QSRVDMP 7-5, 7-26
- QSTRPRTWTR 7-3, 7-19
- QSTRUPPGM 7-3, 7-18
- QSYSLIBL 7-5, 7-22
- QTIME 7-2, 7-9
- QTOTJOB 7-5, 7-24
- QUPSDLYTIM 7-3, 7-19, 7-20
- QUPSMSGQ 7-3, 7-21
- QUSRLIBL 7-5, 7-22
- QYEAR 7-2, 7-8
- retrieving 7-1
- summary 7-1
- system control 7-12
- time 7-9
- using 7-1

system, characteristics of shipped
- description 1-1, 1-7
- summary of effect on a running system 1-17
- use for batch jobs 1-18
- use for interactive jobs 1-10
- use for spooling jobs 1-20
- work management objects 1-10

T

temporary storage, maximum 2-15

Test Request key 2-49

TFRGRPJOB (Transfer to Group Job) command 3-3

TFRJOB (Transfer Job) command
- description of using 2-43
- use of 1-5

time slice 5-9
- changing for duration of a CL command 4-9
- description 2-15

time stamp
- spooled files 2-17

time system values 7-9

time-dependent scheduling 4-11
- ending 4-32
- recovery considerations 4-16
- SBMTIMJOB sample command 4-12, 4-23
- security considerations 4-16
- SNDTIMMSG sample command 4-11, 4-18

trace data
- collecting 6-2
- dumping 6-3

trace table, internal 6-3

Transfer Job (TFRJOB) command
- description of using 2-43
- use of 1-5

Transfer to Group Job (TFRGRPJOB) command 3-3

transferring a job 2-43

tuning
- basic 5-14
- IPL 5-14
- specialized 5-20

U

unattended environment, setting up 4-6

user profile
- QPGMR 1-2
- QSECOFR 1-2
- QSYSOPR 1-2
- QUSER 1-2
- use with work management 1-7
- using initial menu in 4-2
- using initial program in 4-2

user-defined storage pools
- activity level 2-45
- description 2-4

V

validity-checking, job accounting 8-20

W

wait state
- key/think 5-4
- long 5-3
- short 5-3

wait time (DFTWAIT parameter) 2-15

work entry
- autostart job entry 2-7
- description 1-4
- job queue entry 2-7
- work station entry 2-7

work management
- See also performance tuning
- Attention by handling 3-1
- characteristics of the shipped system 1-7
- concepts and terms 1-2
- effect of shipped objects on a running system 1-17
- group jobs 3-1
- introduction 1-1
- objects
  - changes to IBM-shipped 2-3
  - examples of changing 4-4
  - shipped with the system 1-7
  - summary of 2-20
- overview 1-1

work management *(continued)*
- structure 1-2
- summary of concepts and terms 1-2
- using functions
  - creating subsystems 2-47
  - description 2-3
- using shipped objects
  - for programmer sign-on 1-12
  - for work station user to sign on 1-7

work station entry
- activity level 2-45
- adding 2-48
- contents 2-7
- description 1-4

work station type entry
- activity level 2-45
- contents 2-7

work station user
- signing on 1-7

Work with Active Jobs (WRKACTJOB) command 5-13

Work with Job Queue (WRKJOBQ) command 2-50

Work with Output Queue (WRKOUTQ) command 2-50

Work with Subsystem Descriptions (WRKSBSD) command 2-4

Work with System Status (WRKSYSSTS) command 5-10

WRKACTJOB (Work with Active Jobs) command 5-13

WRKJOBQ (Work with Job Queue) command 2-50

WRKOUTQ (Work with Output Queue) command 2-50

WRKSBSD (Work with Subsystem Descriptions) command 2-4

WRKSYSSTS (Work with System Status) command 5-10

AS/400™ Programming:
Work Management
Guide

SC21-8078-0

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name

Company or Organization

Address

City State Zip Code

Phone No.

Area Code

No postage necessary if mailed in the U.S.A.

Fold and Tape **Please do not staple**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 /ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and Tape **Please do not staple**

IBM®

Cut Along Line

AS/400™ Programming:
Work Management
Guide

SC21-8078-0

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name

Company or Organization

Address

City State Zip Code

Phone No.

Area Code

No postage necessary if mailed in the U.S.A.

Fold and Tape **Please do not staple**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 /ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and Tape **Please do not staple**

IBM®

Cut Along Line

AS/400™ Programming:
Work Management
Guide

SC21-8078-0

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your IBM representative or your IBM-approved remarketer to request additional publications.

Name

Company or Organization

Address

City State Zip Code

Phone No.

Area Code

No postage necessary if mailed in the U.S.A.

Fold and Tape **Please do not staple**

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS / PERMIT NO. 40 /ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 245
Rochester, Minnesota, U.S.A. 55901

Fold and Tape **Please do not staple**

IBM®

IBM®

Program Number
5728-SS1

21F2725

SC21-8078-0

Printed in U.S.A.

93X0314

SX21-9926-00

Printed in U.S.A.